SIEMENS

SIMATIC

Systemsoftware für M7-300/400 Installieren und Bedienen

Benutzerhandbuch
Dieses Handbuch ist Bestandteil des Dokumentationspaketes mit der Bestellnummer:
6ES7802–0FA14–8AA0

Vorwort, Inhaltsverzeichnis

Produktübersicht 1

Installieren auf PC/PG 2

Installieren der M7-Zielsysteme 3

Bedienen und überwachen 4

Arbeiten mit Massenspeichern 5

Low-Level-Debugger 6

Ladbare Treiber 7

Anhänge

Betriebssysteme und Leistungsmerkmale A

CLI-Kommandos B

Debugger-Kommandos C

Systemzustandsliste SZL D

Diagnosedaten E

Literaturverzeichnis F

Glossar, Index

C79000-G7000-C850-02

Sicherheitstechnische Hinweise

Dieses Handbuch enthält Hinweise, die Sie zu Ihrer persönlichen Sicherheit sowie zur Vermeidung von Sachschäden beachten müssen. Die Hinweise sind durch ein Warndreieck hervorgehoben und je nach Gefährdungsgradfolgendermaßendargestellt:

Gefahr

bedeutet, daß Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten **werden**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Warnung

bedeutet, daß Tod, schwere Körperverletzung oder erheblicher Sachschaden eintreten **können**, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Vorsicht

bedeutet, daß eine leichte Körperverletzung oder ein Sachschaden eintreten können, wenn die entsprechenden Vorsichtsmaßnahmen nicht getroffen werden.

Hinweis

ist eine wichtige Information über das Produkt, die Handhabung des Produktes oder den jeweiligen Teil der Dokumentation, auf den besonders aufmerksam gemacht werden soll.

Qualifiziertes Personal

Inbetriebsetzung und Betrieb eines Gerätes dürfen nur von **qualifiziertem Personal** vorgenommen werden. Qualifiziertes Personal im Sinne der sicherheitstechnischen Hinweise dieses Handbuchs sind Personen, die die Berechtigung haben, Geräte, Systeme und Stromkreise gemäß den Standards der Sicherheitstechnik in Betrieb zu nehmen, zu erden und zu kennzeichnen.

Bestimmungsgemäßer Gebrauch

Beachten Sie folgendes:

Warnung

Das Gerät darf nur für die im Katalog und in der technischen Beschreibung vorgesehenen Einsatzfälle und nur in Verbindung mit von Siemens empfohlenen bzw. zugelassenen Fremdgeräten und -komponenten verwendet werden.

Der einwandfreie und sichere Betrieb des Produktes setzt sachgemäßen Transport, sachgemäße Lagerung, Aufstellung und Montage sowie sorgfältige Bedienung und Instandhaltung voraus.

Warenzeichen

SIMATIC®, SIMATIC HMI® und SIMATIC NET® sind eingetragene Warenzeichen der SIEMENS AG.

Die übrigen Bezeichnungen in dieser Schrift können Warenzeichen sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Copyright Siemens AG 1998 All rights reserved

Weitergabe sowie Vervielfältigung dieser Unterlage, Verwertung und Mitteilung ihres Inhalts ist nicht gestattet, soweit nicht ausdrücklich zugestanden. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte vorbehalten, insbesondere für den Fall der Patenterteilung oder GM-Eintragung

Siemens AG
Bereich Automatisierungs- und Antriebstechnik
Geschäftsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D- 90327 Nürnberg

Haftungsausschluß

Wir haben den Inhalt der Druckschrift auf Übereinstimmung mit der beschriebenen Hard-und Software geprüft. Dennoch können Abweichungen nicht ausgeschlossen werden, so daß wir für die vollständige Übereinstimmung keine Gewähr übernehmen. Die Angaben in dieser Druckschrift werden regelmäßig überprüft, und notwendige Korrekturen sind in den nachfolgenden Auflagen enthalten. Für Verbesserungsvorschläge sind wir dankbar.

© Siemens AG 1998
Technische Änderungen bleiben vorbehalten.

Siemens Aktiengesellschaft

Bestell-Nr. C79000-G7000-C850

Vorwort

Zweck des Handbuchs

Dieses Handbuch unterstützt Sie bei der Installation und Inbetriebnahme der Automatisierungsrechnerfamilie M7-300 und M7-400 unter dem Betriebssystem M7 RMOS32. Sie erhalten Informationen über

- die Hard– und Softwareumgebung des M7 RMOS32
- die Installation der Systemsoftware auf PC/PG
- die Installation der Systemsoftware auf dem M7–300/400 Zielsystem
- Einstellungen und Konfigurationsmöglichkeiten
- die Bedienung des M7–300/400 über Remote–Terminal
- den Command–Line–Interpreter CLI von M7 RMOS32
- den Low–Level–Debugger von M7 RMOS32

Leserkreis

Dieses Handbuch wendet sich in erster Linie an Inbetriebsetzer von M7–300 und M7–400 Automatisierungssystemen.

Gültigkeitsbereich des Handbuchs

Dieses Handbuch ist gültig für die Automatisierungsrechner M7–300 und M7–400 mit der Systemsoftware M7–SYS RT V 4.0

Was ist neu?

Neu in diesem Handbuch sind folgende Themen zu Funktionsänderungen und Erweiterungen die in der Systemsoftware ab Version V4.0 unterstützt werden sowie zusätzliche Referenzinformationen:

Thema	Kapitel
Freigegebene Betriebssystemkonfigurationen: M7 RMOS32 auf Festplatte und M7 RMOS32 mit MS-DOS auf Memory Card	3.4, 3.5
Ändern von Konfigurationsdateien im SIMATIC Manager	3.9
Kommunikation über Industrial Ethernet (TCP/IP)	A.1
Kommando PING für Industrial Ethernet	B.22

Einordnung in die Informationslandschaft

Die Systemsoftware für Automatisierungsrechner M7–300 und M7–400 mit M7-SYS RT wird in drei Handbüchern dokumentiert, die ein Dokumentationspaket bilden, das getrennt vom Softwarepaket M7–SYS RT bestellt werden kann. Die Handbücher sind in der folgenden Tabelle aufgelistet.

Handbuch	Inhalt
Systemsoftware für M7–300/400 Installieren und Bedienen, Benutzerhandbuch	Installation und Bedienung von Automatisierungsrechnern M7–300/400.
Systemsoftware für M7–300/400 Programmentwurf, Programmierhandbuch	Entwerfen und Erstellen von C/C++–Programmen.
Systemsoftware für M7–300/400 System– und Standardfunktionen, Referenzhandbuch	Detailinformationen zur Programmierung mit M7–SYS

Rückmeldung zur Dokumentation

Um Ihnen und zukünftigen M7-SYS RT Anwendern eine optimale Dokumentation anbieten zu können, bitten wir Sie, uns hierbei zu unterstützen. Bei Anmerkungen zum vorliegenden *Handbuch* füllen Sie bitte den Fragebogen am Ende des Handbuchs aus und schicken Sie ihn an die dort angegebene Adresse. Bitte tragen Sie dort auch Ihre persönliche Bewertung ein.

Literaturhinweise /.../

Hinweise auf weitere Dokumentationen sind mit Hilfe von Literaturnummern in Schrägstrichen /.../ angegeben. Anhand dieser Nummern können Sie dem Literaturverzeichnis am Ende des Handbuchs den Titel der Dokumentation entnehmen.

SIMATIC Customer Support Hotline

Weltweit erreichbar zu jeder Tageszeit:

Nürnberg

SIMATIC BASIC Hotline

Ortszeit: Mo.-Fr. 8:00 bis 18:00
Telefon: +49 (911) 895-7000
Fax: +49 (911) 895-7002
E-Mail: simatic.support@nbgm.siemens.de

SIMATIC Premium Hotline

(kostenpflichtig, nur mit SIMATIC Card)
Zeit: Mo.-Fr. 0:00 bis 24:00
Telefon: +49 (911) 895-7777
Fax: +49 (911) 895-7001

Johnson City

SIMATIC BASIC Hotline

Ortszeit: Mo.-Fr. 8:00 bis 17:00
Telefon: +1 423 461-2522
Fax: +1 423 461-2231
E-Mail: simatic.hotline@sea.siemens.com

Singapur

SIMATIC BASIC Hotline

Ortszeit: Mo.-Fr. 8:30 bis 17:30
Telefon: +65 740-7000
Fax: +65 740-7001
E-Mail: simatic@singnet.com.sg

SIMATIC Customer Support Online-Dienste

Das SIMATIC Customer Support bietet Ihnen über die Online–Dienste umfangreiche zusätzliche Informationen zu den SIMATIC–Produkten:

- Allgemeine aktuelle Informationen erhalten Sie
  - im **Internet** unter `http://www.ad.siemens.de/simatic`
  - über **Fax-Polling** Nr. 08765-93 02 77 95 00
- Aktuelle Produkt–Informationen und Downloads, die beim Einsatz nützlich sein können:
  - im **Internet** unter `http://www.ad.siemens.de/support/html–00/`
  - über das **Bulletin Board System** (BBS) in Nürnberg (*SIMATIC Customer Support Mailbox)* unter der Nummer +49 (911) 895-7100.

    Verwenden Sie zur Anwahl der Mailbox ein Modem mit bis zu V.34 (28,8 kBaud), dessen Parameter Sie wie folgt einstellen: 8, N, 1, ANSI, oder wählen Sie sich per ISDN (x.75, 64 kBit) ein.

SIMATIC Trainingscenter

Um Ihnen den Einstieg in die Automatisierungssysteme SIMATIC S7 und M7 zu erleichtern, bieten wir entsprechende Kurse an. Wenden Sie sich bitte an Ihr regionales Trainingscenter oder an das zentrale Trainingscenter in D-90327 Nürnberg. Tel. 0911/895 3154.

Weitere Unterstützung

Bei weiteren Fragen zu den SIMATIC Produkten wenden Sie sich bitte an Ihre Siemens-Ansprechpartner in den für Sie zuständigen Vertretungen und Geschäftsstellen. Die Adressen finden Sie in Katalogen und in Compuserve (`go autforum`).

Inhaltsverzeichnis

1 Produktübersicht 1-1

1.1 M7-Optionssoftware 1-2

1.2 Betriebssysteme für M7-300/400 1-5

1.3 M7 RMOS32 - Kurzbeschreibung 1-7

1.4 M7 RMOS32 mit MS-DOS - Kurzbeschreibung 1-8

2 Installieren auf PC/PG 2-1

2.1 Installieren von M7-SYS RT V4.0 2-1

2.2 Installieren mehrerer Versionen von M7-SYS 2-3

3 Installieren der M7-Zielsysteme 3-1

3.1 Allgemeines zur Installation 3-2

3.2 Datensicherheit bei Netzausfall 3-7

3.3 Installieren von M7 RMOS32 auf Memory Card 3-8

3.4 Installieren von M7 RMOS32 mit MS-DOS auf Memory Card 3-10

3.5 Installieren von M7 RMOS32 auf Festplatte 3-12

3.6 Installieren von M7 RMOS32 mit MS-DOS auf Festplatte 3-14

3.7 Nachinstallation des M7-Betriebssystems 3-15

3.8 Aktualisieren der Firmware 3-17

3.9 Ändern der Konfigurationsdateien 3-20

3.10 Die Datei RMOS.INI 3-24

3.11 Die Datei INITTAB 3-27

4 Bedienen und überwachen 4-1

4.1 Bedienen über Remote Terminal 4-1

4.2 Arbeiten mit dem Kommandozeileninterpreter CLI 4-4

4.3 Programme auf M7-Zielsystem übertragen und löschen 4-10

4.4 Starten der Anwendungsprogramme auf dem M7-Zielsystem 4-16

4.5 Abfrage- und Steuerfunktionen für M7-300/400 4-17

5 Arbeiten mit Massenspeichern 5-1

5.1 Formatieren der Memory Cards und OSD 5-2

5.2 Formatieren der Festplatte und Diskette 5-3

5.3 Festplattenpartitionierungsprogramm HDPART 5-4

	5.4	Formatierung von Memory Cards und OSDs unter MS-DOS	5-9
	5.5	Arbeiten mit Memory Cards	5-11
	5.6	Programm REMAP_A	5-12
6	**Low-Level-Debugger**		**6-1**
	6.1	Betriebsarten Task-Modus und Monitor-Modus	6-2
	6.2	Bedienung des Debuggers	6-5
	6.3	Allgemeine Syntax-Regeln	6-7
	6.4	Kurzbeschreibung der Debugger-Kommandos	6-9
7	**Ladbare Treiber**		**7-1**
	7.1	Was Sie über ladbare Treiber wissen sollten	7-1
	7.2	Laden eines Treibers	7-3
A	**Betriebssysteme und Leistungsmerkmale**		**A-1**
	A.1	Leistungsmerkmale der Zentralbaugruppen und Applikationsbaugruppen	A-2
	A.2	M7 RMOS32 - Konfigurierung	A-5
	A.3	Belegung des Hauptspeichers	A-9
	A.4	M7 RMOS32 für DOS - Konfigurierung	A-10
	A.5	Komponenten von M7 RMOS32-DOS	A-15
	A.5.1	RM3PMEM.SYS	A-16
	A.5.2	RM3RESET.SYS	A-16
	A.6	Besonderheiten unter M7 RMOS32-DOS	A-17
B	**CLI–Kommandos**		**B-1**
	B.1	BYT8250	B-3
	B.2	CANCEL	B-5
	B.3	CD	B-6
	B.4	CHGKBD	B-7
	B.5	COPY	B-8
	B.6	CPRI	B-10
	B.7	DATE	B-11
	B.8	DEL	B-12
	B.9	DEVICE	B-13
	B.10	DIR	B-16
	B.11	DISMOUNT	B-17
	B.12	ECHO	B-18
	B.13	ERROR	B-19
	B.14	EXIT	B-20
	B.15	FORMAT	B-21

B.16 FTLFORM B-22
B.17 HELP B-24
B.18 MD B-25
B.19 MOUNT B-26
B.20 NPX – nicht für weitere Entwicklungen B-27
B.21 PATH B-28
B.22 PING B-30
B.23 PROMPT B-31
B.24 RD B-32
B.25 RDISK B-33
B.26 RENAME B-34
B.27 SCANDISK B-35
B.28 SESSION B-37
B.29 SET B-38
B.30 START B-39
B.31 SYSTAT B-40
B.32 TIME B-41
B.33 VER B-42
B.34 CLI-Fehlermeldungen B-43

C Debugger-Kommandos C-1

C.1 Syntax der Debugger-Kommandos C-3
C.2 ASM C-7
C.3 BASE C-9
C.4 BREAKS C-10
C.5 CALCULATE C-11
C.6 CALL C-12
C.7 CHANGE C-13
C.8 CONT C-14
C.9 CPUREG C-15
C.10 DIR C-16
C.11 DISPLAY C-19
C.12 EVALUATE C-20
C.13 EXIT C-21
C.14 EXITK C-22
C.15 FILL C-23
C.16 FREETASK C-24

C.17 GO C-25
C.18 HALT C-26
C.19 HELP C-27
C.20 IN C-28
C.21 INHIB C-29
C.22 KILL C-30
C.23 LINES C-31
C.24 LOADTASK C-32
C.25 MONITOR C-34
C.26 OUT C-35
C.27 QUALIFY C-36
C.28 QUERY C-37
C.29 REGS C-39
C.30 REPORT C-41
C.31 SET C-47
C.32 STACK C-49
C.33 START C-50
C.34 STEP C-51
C.35 SVC C-52
C.36 SWITCH C-54
C.37 TASK C-55
C.38 TCB C-56
C.39 TCD C-57
C.40 Fehlermeldungen des Debuggers C-58

D Systemzustandsliste SZL D-1

D.1 Überblick über die Systemzustandsliste (SZL) D-2
D.2 Aufbau einer SZL-Teilliste D-3
D.3 SZL-ID D-4
D.4 Mögliche SZL-Teillisten D-5
D.5 SZL-ID W#16#xy11 - Baugruppen-Identifikation D-6
D.6 SZL-ID W#16#xy12 - CPU-Merkmale D-7
D.7 SZL-ID W#16#xy13 - Anwenderspeicherbereiche D-9
D.8 SZL-ID W#16#xy14 - Systembereiche D-10
D.9 SZL-ID W#16#xy15 - Bausteintypen D-11
D.10 SZL-ID W#16#xy22 - Alarmstatus D-13
D.11 SZL-ID W#16#xy24 - Betriebszustand und Betriebszustandsübergang . . D-14

D.12 SZL-ID W#16#xy32 - Kommunikationszustandsdaten D-18
D.13 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0001 . . . . . D-19
D.14 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0005 . . . . . D-20
D.15 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0008 . . . . . D-21
D.16 SZL-ID W#16#xy91 - Baugruppenzustandsinformation D-22
D.17 SZL-ID W#16#xy92 - Baugruppenträger-/Stationszustandsinformation . . D-24
D.18 SZL-ID W#16#xyA0 - Diagnosepuffer D-26
D.19 SZL-ID W#16#00B1 - Baugruppendiagnoseinfo D-27
D.20 SZL-ID W#16#00B2 - Baugruppendiagnosedaten über geographische Adresse D-28
D.21 SZL-ID W#16#00B3 - Baugruppendiagnosedaten über logische Basisadresse D-29
D.22 SZL-ID W#16#00B4 – Diagnosedaten eines DP-Slaves D-30

E Diagnosedaten E-1

F Literaturverzeichnis F-1

Glossar

Index

Produktübersicht

1

Übersicht

Mit STEP 7 und der M7-Optionssoftware können Sie Hochsprachen wie C oder C++ sowie grafische Programmiersoftware wie CFC (Continuous Funktion Chart) einsetzen, um Anwendungsprogramme für den Automatisierungsrechner M7-300/400 zu erstellen.

In diesem Kapitel erfahren Sie, welche Möglichkeiten Sie haben, Anwendungsprogramme für den Automatisierungsrechner M7-300/400 zu erstellen. Zudem erhalten sie einen Überblick über die Systemsoftware für M7-300/400.

M7-Optionssoftware

Für die Erstellung der Programme benötigen Sie zusätzlich zu STEP 7 die Systemsoftware für M7-300/400 sowie eine Entwicklungsumgebung für M7-Programme (ProC/C++ oder CFC). Diese Softwarekomponenten sind in den nachfolgenden Kapiteln beschrieben.

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
1.1	M7-Optionssoftware	1-2
1.2	Betriebssysteme für M7-300/400	1-5
1.3	M7 RMOS32 - Kurzbeschreibung	1-7
1.4	M7 RMOS32 mit MS-DOS - Kurzbeschreibung	1-8

1.1 M7-Optionssoftware

Überblick

STEP 7 bietet Ihnen die Grundfunktionalität an, die Sie benötigen um:

- Projekte zu erzeugen und zu verwalten
- Zielsystem-Hardware zu konfigurieren und zu parametrieren
- Netzwerke und Verbindungen zu projektieren
- Symboldaten zu verwalten

Diese Funktionalität ist unabhängig davon, ob Ihr Zielsystem SIMATIC S7 oder SIMATIC M7 ist.

Die Zielsysteme SIMATIC S7 oder SIMATIC M7 wirken sich durch die unterschiedlichen Betriebssysteme und Ablaufsoftware hauptsächlich auf die Programmierung der Anwendungen aus.

Für die Erstellung von M7-Anwendungsprogrammen benötigen Sie zusätzlich zu STEP 7 noch M7-Optionssoftware.

Tabelle 1-1 Optionssoftware für die M7-Programmierung

Software	Inhalt
M7-SYS RT	• Betriebssystem M7 RMOS32 • Systembibliothek M7-API • MPI-Unterstützung
CFC für S7 und M7	Programmiersoftware für CFC-Anwendungen (CFC = Continuous Function Chart)
M7-ProC/C++	• Einbindung der Borland-Entwicklungsumgebung in STEP 7 • Symbolimport-Editor und -Generator • Organon-Debugger xdb386
Borland C++	Borland C++-Entwicklungsumgebung

Zusammen mit der M7-Optionssoftware unterstützt Sie STEP 7 zusätzlich bei folgenden Tätigkeiten:

- Daten auf das Zielsystem über MPI übertragen
- Informationen über das Zielsystem abfragen
- bestimmte Einstellungen auf dem Zielsystem vornehmen und das Zielsystem urlöschen

Abhängigkeiten

Das folgende Bild zeigt die Abhängigkeiten der M7-Optionssoftware:

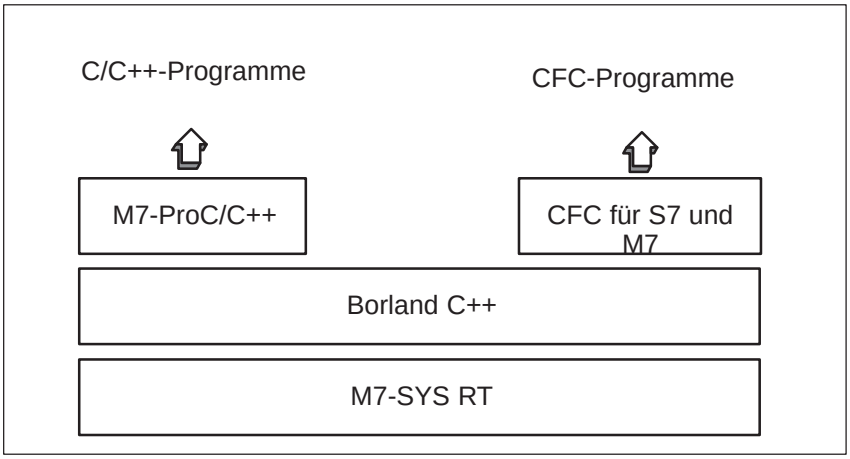

Bild 1-1 Abhängigkeiten der M7-Optionssoftware für M7-Programmierung

Tabelle 1-2 Zusammenfassung

Für die Erstellung von ...	benötigen Sie die M7-Optionssoftware ...
C/C++-Programmen	1. M7-SYS RT 2. M7-ProC/C++ 3. Borland C++
CFC-Programmen	1. M7-SYS RT 2. CFC für S7 und M7 3. Borland C++

Wo Sie Unterstützung finden

Die spezifischen Werkzeuge für die Erstellung von M7-Anwendungen sind teilweise in STEP 7 und teilweise in der M7-Optionssoftware integriert.

Aus der folgenden Tabelle erfahren Sie, welche Unterstützung Sie in den einzelnen Softwarepaketen finden:

Tabelle 1-3 Unterstützung bei der Erstellung von M7-Anwendungsprogrammen

Software ...	**Unterstützt Sie ...**
STEP 7	• beim Installieren des M7-Betriebssystems, • der Verwaltung des M7-Zielsystems, • dem Übertragen, Starten und Löschen der M7-Anwendungsprogramme • dem Abrufen von Zustands-und Diagnosedaten • beim Urlöschen der CPU
M7-SYS RT	durch die Dienste des M7-Betriebssystems und der -Systemsoftware bei: • Programmablaufsteuerung, • Speicher- und Ressourcenverwaltung, • Zugriff auf Rechner- und SIMATIC-Hardware, • Alarmverwaltung, • Diagnose, • Zustandsüberwachung und • Kommunikation
M7-ProC/C++	• durch integrierte Code-Erzeugung (Integration der Borland-Entwicklungsumgebung in STEP 7) • durch Einbindung der Projekt-Symbolik in den Quellcode und • durch integrierte Debug-Funktionalität
Borland C++	• beim Erstellen von C- und C++-Programmen
CFC für S7 und M7	• beim Erstellen, Testen und Debuggen von CFC-Programmen und • beim Starten und Ablaufen von CFC-Programmen

1.2 Betriebssysteme für M7-300/400

Einführung

Durch seine standardisierte PC-Architektur bildet der Automatisierungsrechner M7-300/400 eine freiprogrammierbare Erweiterung der SIMATIC-Automatisierungsplattform. Die Anwendungsprogramme für SIMATIC M7 können Sie in einer Hochsprache wie C oder auch grafisch mit CFC programmieren.

Für Anwendungen, die in den Hochsprachen C und C++ erstellt werden, sind die Dienste des Betriebssystems von entscheidender Bedeutung. Das Betriebssystem übernimmt für die Anwendung folgende Aufgaben:

- Zugriff auf die Hardware
- Verwaltung der Ressourcen
- Systemeinbindung
- Kommunikation zu anderen Komponenten im System

Echtzeitbetriebssystem

Für die Lösung von Automatisierungsaufgaben setzen wir auf dem Automatisierungsrechner SIMATIC M7 das Echtzeitbetriebssystem M7 RMOS32 (RMOS=**R**e-altime-**M**ultitasking-**O**perating-**S**ystem) ein. Für die Einbindung in das SIMATIC-System ist M7 RMOS32 um eine Aufrufschnittstelle, das M7-API (API = **A**pplication **P**rogramming **I**nterface) erweitert.

Betriebssystem-Konfigurationen für M7

Das Echtzeitbetriebssystem M7 RMOS32 wird für 32-Bit-Anwendungen zur Lösung zeitkritischer Echtzeit- und Multitasking-Aufgaben eingesetzt. Es ist für M7-Baugruppen in folgenden Konfigurationen verfügbar:

- M7 RMOS32
- M7 RMOS32 mit MS-DOS

Welche Betriebssystem-Konfiguration Sie für Ihr M7-Zielsystem auswählen hängt von den M7-Baugruppen, die Sie einsetzen, ab (siehe Tabelle 1-4):

Tabelle 1-4 Software-Hardware-Konfigurationen

Betriebssystem-Konfiguration	Baugruppe / Hauptspeicher	PROFIBUS-DP und TCP/IP Ja/Nein	Installation auf Massenspeicher
M7 RMOS32	FM 356-4 / 4MB	nein	Memory Card
	FM 356-4 / 8MB	ja	$\geq$ 4 MB
	CPU 388-4 / 8MB	ja	oder Festplatte
	FM 456-4 / 16MB	ja	
	CPU 488-3 / 16MB	ja	
	CPU 486-3 / 16MB	ja	
M7 RMOS32 mit MS-DOS	FM 356-4 / 8MB	nein	Memory Card
	CPU 388-4 / 8MB	nein	$\geq$ 4 MB
	FM 456-4 / 16MB	ja	oder Festplatte
	CPU 488-3 / 16MB	ja	
	CPU 486-3 / 16MB	ja	

Hardware-Konfigurationen mit PROFIBUS-DP und TCP/IP werden nur mit folgenden Betriebssystemen unterstützt:

- M7 RMOS32 mit Hauptspeicher von mindestens 8 MB
- M7 RMOS32 mit MS-DOS mit Hauptspeicher von 16 MB

Zusätzliche Hardware

M7 RMOS32 mit MS-DOS ist grundsätzlich nur auf M7-Baugruppen einsetzbar, die über folgende zusätzliche Hardware verfügen: VGA-Bildschirm und Tastatur über das Schnittstellenmodul IF962-VGA.

Massenspeicher

M7-Zentralbaugruppen und M7-Applikationsbaugruppen verfügen über folgende Arten von Massenspeichern (siehe Tabelle 1-5):

- Memory Cards (wie auch S7-CPU)
- Festplatte und Diskette

  Alle programmierbaren M7-Baugruppen können wahlweise mittels der MSM-Erweiterungsbaugruppen mit einer Festplatte und einem 3,5''-Diskettenlaufwerk ausgerüstet werden. Die Diskette können Sie, wie die Memory Card, sowohl am PC/PG als auch am M7-Zielsystem ansprechen.
- Onboard Silicon Disk (OSD)
  Dieser Massenspeicher verhält sich wie ein Festplattenlaufwerk, auf dem Anwendungsprogramme gespeichert werden können. Die FM 456-4 kann damit optional ausgestattet werden.

Tabelle 1-5 Massenspeicher bei M7-Zielsystemen

Massenspeicher	Größe	M7-300 Baugruppe	M7-400 Baugruppe
Festplatte	ab 512MB	MSM378	MSM478
Diskette 3,5”	1,44MB	MSM378	MSM478
Memory Card	2*), 4, 8, 16 MB	CPU388-4 FM356-4	CPU488/486-3 FM456-4
OSD	4MB	-	optional in FM456-4

*) nicht für ein vollständiges Betriebssystem

Hinweis

Welche Systemvarianten bezüglich Hauptspeicherausbau und Massenspeicher von der aktuellen Produktversion unterstützt werden, entnehmen Sie der Tabelle 1-4 auf Seite 1-6.

1.3 M7 RMOS32 - Kurzbeschreibung

Eigenschaften

M7 RMOS32 hat folgende Merkmale:

- vorkonfigurierte Betriebssystemvariante für M7-300/400 CPU und FM
- alleinige Kontrolle der Hardware durch M7 RMOS32
- definierte Reaktionszeiten im Mikro- bzw. Millisekundenbereich, Echtzeitfähigkeit für Messen, Steuern, Regeln

Gerätesteuerung

Folgende Geräte können von M7RMOS32 gesteuert werden:

- Vier M7 RMOS32-Konsolen über EGA/VGA
- Je eine M7 RMOS32-Konsole an COM1 und COM2
- Ein Drucker an LPT1 (Centronics-Schnittstelle)
- Eine Memory Card
- Eine Onboard Silicon Disk (OSD)
- Massenspeicherbaugruppe MSM 378/478 mit einer Festplatte und einem Diskettenlaufwerk 1,44MB

Interrupts

Alle Interrupts werden von M7 RMOS32 behandelt.

Dateiverwaltungssystem

Alle Laufwerke (Disketten, Festplatte, Memory Card und OSD) werden vom RMOS-Dateiverwaltungssystem HSFS (High Speed File System) verwaltet. Eine automatische Festplattenerkennung integriert die Festplatten in das HSFS.

Kommandozeileninterpreter (CLI)

Ähnlich dem DOS-Kommandozeileninterpreter mit Kommandos für M7 RMOS32. Der CLI wird über die M7 RMOS32-Konsole bedient.

Low-Level-Debugger

Integrierter Low-Level-Debugger. Der Debugger wird über die M7 RMOS32-Konsole bedient.

1.4 M7 RMOS32 mit MS-DOS - Kurzbeschreibung

Eigenschaften

M7 RMOS32 für MS-DOS hat folgende Merkmale:

- vorkonfiguriertes Echtzeit-Multitasking-Betriebssystem für PC-kompatible Systeme
- MS-DOS V6.22 läuft als M7 RMOS32-Task
- definierte Reaktionszeiten im Mikro- bzw. Millisekundenbereich, Echtzeitfähigkeit für Messen, Steuern, Regeln

Bootvorgang

MS-DOS wird immer zuerst gebootet. M7 RMOS32 wird mit einem speziellen Ladeprogramm von MS-DOS aus gestartet.

Gerätesteuerung

Die Grafikkarte und die Tastatur werden von MS-DOS gesteuert.

Alle Laufwerke (Disketten, Festplatte usw.) werden, sofern sie dem BIOS bekannt sind, von MS-DOS gesteuert.

Die Interruptcontroller und der Timer-Baustein werden von M7 RMOS32 gesteuert. Die für MS-DOS notwendigen Funktionen werden simuliert.

Andere Geräte und Schnittstellen können wahlweise von MS-DOS oder M7 RMOS32 gesteuert werden. Dies wird bei der Konfigurierung festgelegt.

Interruptbehandlung

Der Timer-Interrupt wird von M7 RMOS32 behandelt. Der Timer-Interrupt für MS-DOS wird von M7 RMOS32 für MS-DOS simuliert.

Die Interrupts von Tastatur, Festplatte und Diskette werden von MS-DOS behandelt.

Alle andere Interrupts können wahlweise von MS-DOS oder M7 RMOS32 behandelt werden. Dies wird bei der Konfigurierung festgelegt.

Dateiverwaltungssystem

Es wird das MS-DOS-eigene Dateiverwaltungssystem verwendet.

Die Datenträger werden für M7 RMOS32 zugänglich gemacht.

Speicherbelegung

Der Speicherbereich von Adresse 0 bis 10FFFFH wird grundsätzlich für MS-DOS reserviert.

Der Speicherbereich ab der Adresse 110000H wird beim Bootvorgang M7 RMOS32 zugeteilt.

Monitor, Tastatur

Monitor und Tastatur können unter MS-DOS mittels Hotkey-Umschaltung <Ctrl>+<Esc> abwechselnd MS-DOS oder M7 RMOS32 zugeordnet werden. Unter M7 RMOS32 können Tastatur und Bildschirm mit <F1>...<F4> vier unterschiedlichen Konsolen zugeordnet werden.

Zusammenspiel des M7 RMOS32-Multitasking mit MS-DOS

Im Gegensatz zum DOS-Dateiverwaltungssystem erlaubt M7 RMOS32 auch quasi-parallele Datei-Operationen, z.B. gleichzeitiges Lesen/Schreiben von Dateien auf Diskette/Festplatte. Dies hat u.a. den Vorteil, daß während des Formatierens einer Diskette parallele Dateioperationen von Echtzeit-Tasks auf anderen Massenspeichern (Festplatte, Memorycard) möglich sind.

Hinweis

Bei M7 RMOS32 werden die nicht-multitaskingfähigen DOS-Aufrufe verriegelt und dadurch die Anforderungen hintereinander abgearbeitet. Für Massenspeicher-Operationen bedeutet dies, daß z.B. während des Formatierens einer Diskette (von DOS aus) Dateioperationen von Echtzeit-Tasks angehalten werden.

Bitte beachten Sie, daß während der Ausführung von Echtzeit-Tasks keine Dateizugriffe unter MS-DOS stattfinden dürfen.

Warmstart

Ein Warmstart der MS-DOS-Task kann mit der Tastenkombination <Ctrl>+<Alt>+<Del> ausgelöst werden. Beim Warmstart wird nur MS-DOS und nicht M7 RMOS32 gebootet.

Während des Warmstarts können gezielt M7 RMOS32-Tasks gestartet werden. Diese Tasks müssen vorher über spezielle Funktionen an- bzw. abgemeldet werden.

Fehlerbehandlung

Privilegverletzungen in der MS-DOS-Task werden von M7 RMOS32 abgefangen. Das laufende MS-DOS-Programm wird dabei automatisch abgebrochen.

Illegale E/A-Befehle, d.h. Zugriffe auf ein von M7 RMOS32 gesteuertes Gerät, werden von M7 RMOS32 abgefangen.

Installieren auf PC/PG

2

Übersicht

In diesem Kapitel finden Sie die Informationen, die Sie für eine erfolgreiche Installation der Optionssoftware M7-SYS RT benötigen.

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
2.1	Installieren von M7-SYS RT V4.0	2-1
2.2	Installieren mehrerer Versionen von M7-SYS	2-3

2.1 Installieren von M7-SYS RT V4.0

Übersicht

Das Softwarepaket M7-SYS RT wird auf PC/PG installiert. Es enthält ein Setup-Programm, das die Installation automatisch durchführt. Während des gesamten Installationsvorgangs werden Sie Schritt für Schritt durch Eingabeaufforderungen auf dem Bildschirm geführt.

Hardwarevoraussetzungen

Um die M7-SYS RT-Software auf PC/PG zu installieren, müssen folgende Hardwarevoraussetzungen erfüllt sein:

- Freien Speicher auf der Festplatte des PC/PG von insgesamt ca. 10 MB
- Freien Speicher von mindestens 1 MB auf dem Laufwerk C: für das Setup-Programm. Die temporären Setup-Dateien werden nach erfolgreicher Installation wieder gelöscht.

Softwarevoraussetzungen

Für die Installation von M7-SYS RT auf PC/PG müssen folgende Softwarevoraussetzungen erfüllt sein:

- Das Betriebssystem **Windows 95** oder **Windows NT** muß auf dem PC/PG installiert sein.
- Die Basissoftware **STEP 7 V4.02** muß auf dem PC/PG installiert sein. M7-SYS RT V4.0 kann auch mit STEP 7 V3.2 eingesetzt werden, dann jedoch ohne den neuen Funktionen. Die Installation von STEP 7 ist in dem STEP 7 Benutzerhandbuch beschrieben.

Für die Installation unter Windows NT müssen Sie Administrationsrechte besitzen.

Vorgehensweise

1. Schließen Sie alle STEP 7 Applikationen
2. Legen Sie die erste Diskette in das Laufwerk des PC/PG ein und starten Sie das Programm *Setup.exe* von M7-SYS RT. Daraufhin werden die Dateien von M7-SYS RT auf PC/PG kopiert und Einträge in MS-Windows-Dateien gemacht. Wichtige Bedienungshinweise werden Ihnen im Laufe des Setups gegeben.
3. Am Ende wird eine Meldungsbox aufgeblendet, die Ihnen den erfolgreichen Abschluß der Installation anzeigt.
4. Bevor Sie M7-SYS RT nutzen, muß Ihr PC/PG neu gestartet werden. Erst dann werden alle Einstellungen wirksam.

Vorsicht

M7-SYS RT registriert sich in MS-Windows 95/NT-Systemdateien. Sie dürfen keine M7-SYS RT-Dateien und -Ordner verschieben oder M7-SYS RT-Daten in der MS-Windows-Registry modifizieren. Ein korrekter Programmablauf kann nach solchen Modifikationen nicht mehr möglich sein.

Deinstallation

Deinstallieren Sie M7-SYS RT – wie in Windows 95/NT üblich – über die Systemsteuerung.

Hinweis

Wenn Sie mehrere Versionen von M7-SYS installiert haben, müssen Sie auch alle deinstallieren. In diesem Fall kann eine Version von M7-SYS nicht einzeln deinstalliert werden.

Beachten Sie, daß bei der Deinstallation von M7-SYS vom PC/PG Einträge, die in der Datei WINSTART.BAT durch das System vorgenommen wurden, nicht gelöscht werden.

2.2 Installieren mehrerer Versionen von M7-SYS

Mehrere Versionen von M7-SYS

In STEP 7 ist es möglich, mehrere Versionen der Systemsoftware M7-SYS gleichzeitig auf dem PC/PG zu betreiben. Daraus ergeben sich folgende Kompatibilitätsaussagen:

- Installationsreihenfolge:
  Bei der Installation auf PC/PG müssen Sie immer die neuere Version **nach** der älteren installieren (z.B. M7-SYS RT V4.0 nach M7-SYS V2.0), um zu verhindern, daß neuere gemeinsame Komponenten durch ältere überschrieben werden.
- Deinstallation:
  Wegen der gemeinsamen gemeinsamen Komponenten ist die vollständige Deinstallation einzelner Versionen nicht möglich. Wenn mehrere Versionen auf dem PC/PG installiert sind, müssen Sie zusammen deinstalliert werden.
- Abwärtskompatibilität der Programme - wird nicht unterstützt.
  D.h. Programme, die mit M7-SYS RT V4.0 übersetzt und gebunden wurden, können nicht auf einem Zielsystem mit dem Betriebssystem von V2.0 oder V1.2 ablaufen.
- Aufwärtskompatibilität der Programme - wird mit Ausnahme von Windows 3.11-Programmen unterstützt.
  D.h. Programme, die mit M7-SYS V2.0 oder V1.2 übersetzt und gebunden wurden, können auf einem Zielsystem mit dem Betriebssystem von V4.0 (M7 RMOS32 evtl. mit MS-DOS) ablaufen.

Auswählen der Betriebssystemversion für das Zielsystem

Bei der Installation des Betriebssystems auf eine M7-CPU oder -FM können Sie im SIMATIC Manager (Menübefehl **Zielsystem > M7-Zielsystem verwalten**) die gewünschte Version auswählen.

Umstieg von V1.2 auf V2.0 oder V4.0

Wenn Sie eine CPU 488-4/5 durch eine CPU 486/8-3 austauschen, müssen Sie von V1.2 auf V2.0 oder M7-SYS RT V4.0 umsteigen.

Installieren der M7-Zielsysteme

3

Überblick

In diesem Kapitel erfahren Sie:-

- wie Sie die Installation vorbereiten
- wie Sie ein Betriebssystem auf dem M7-Zielsystem installieren und nachinstallieren
- wie Sie die Firmware aktualisieren
- welche Konfigurierungsmöglichkeiten Sie für M7 RMOS32 haben.

Menübefehl

Die Installation auf dem M7-Zielsystem wird über den SIMATIC Manager durchgeführt.

Um in die M7-Zielsystemverwaltung zu gelangen, rufen Sie aus dem Kontext eines Projektes, das Stationen mit M7-CPUs oder -FMs enthält, mit angewähltem ”M7-Programm”, folgenden Menübefehl auf:

Zielsystem ▸ M7-Zielsystem verwalten

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
3.1	Allgemeines zur Installation	3-2
3.2	Datensicherheit bei Netzausfall	3-7
3.3	Installieren von M7 RMOS32 auf Memory Card	3-8
3.4	Installieren von M7 RMOS32 mit MS-DOS auf Memory Card	3-10
3.5	Installieren von M7 RMOS32 auf Festplatte	3-12
3.6	Installieren von M7 RMOS32 mit MS-DOS auf Festplatte	3-14
3.7	Nachinstallation des M7-Betriebssystems	3-15
3.8	Aktualisieren der Firmware	3-17
3.9	Ändern der Konfigurationsdateien	3-20
3.10	Die Datei RMOS.INI	3-24
3.11	Die Datei INITTAB	3-27

3.1 Allgemeines zur Installation

Zweck der Installation

Zweck der Installation ist, eine vollständige Betriebssystem-Konfiguration einschließlich M7-Systemsoftware auf das **Zielmedium**, dem Massenspeicher eines M7-Systems zu übertragen.

Dieser Abschnitt gibt Ihnen einen Überblick über die Installationsmöglichkeiten und die prinzipielle Vorgehensweise. Schrittweise Installationsanleitungen finden Sie in den nachfolgenden Abschnitten sowie in der Online-Hilfe zur M7-Zielsystemverwaltung.

Installationsmöglichkeiten

Abhängig von dem Massenspeicher des M7-Zielsystems, unterscheidet man zwischen:

1. Installation auf Festplatte. Bei der ersten Installation ist auf dem M7-Zielsystem noch kein lauffähiges Betriebssystem installiert und es ist noch keine MPI-Verbindung möglich.
2. Installation auf Memory Card. Auf einer Memory Card findet ein vollständiges M7 RMOS32-Betriebssystem mit Anwendungsprogrammen Platz.

Prinzipielles Vorgehen

Um ein Betriebssystem zu installieren:

1. Selektieren Sie das Objekt ”M7-Programm” in Ihrem Projekt.
2. Rufen Sie den Menübefehl **Zielsystem ▸ M7-Zielsystem verwalten** auf.
3. Öffnen Sie den Register-Dialog ”BS Installieren”
4. Treffen Sie folgende Auswahlen (siehe Bild auf Seite 3-3):
   - Betriebssystem-Konfiguration
   - Version des Betriebssystems im PG (nur wenn Sie mehrere Versionen von M7-SYS auf dem PG/PC installiert haben)
   - Medium
   - Lokales Laufwerk (nur auf PC/PG mit Windows 95) und Partner Laufwerk, wenn Sie das Medium ”MPI/RFS” benutzen
5. Betätigen Sie die Schaltfläche ”Installieren”.

Alle weiteren Tätigkeiten hängen von der ausgewählten Betriebssystem-Konfiguration und von dem Zielmedium ab.

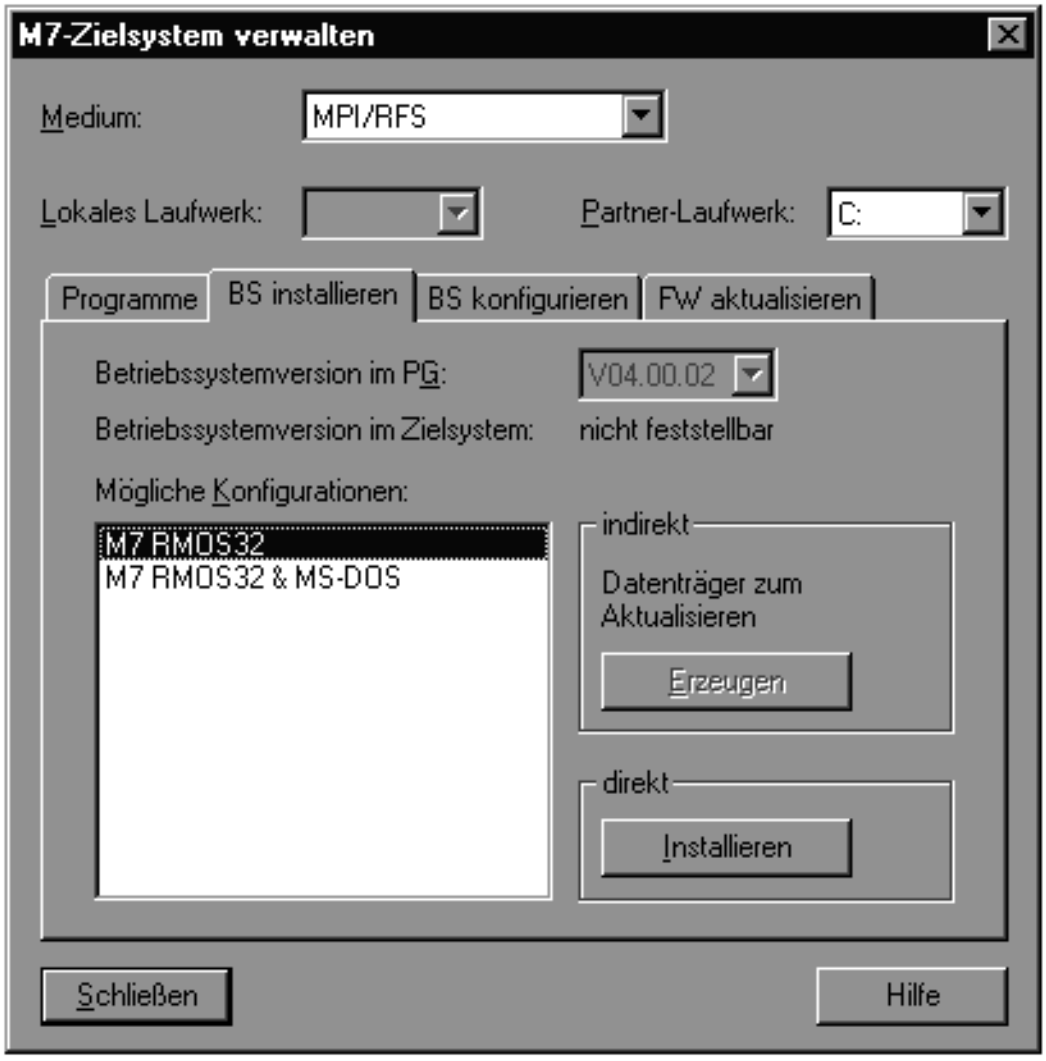

Bild 3-1 Register ”BS Installieren” (STEP 7 auf PC/PG mit Windows NT)

Betriebssystemversion

Im PG:
Wählen Sie hier die Version des Betriebssystems, das Sie auf Ihrem M7-Zielsystem installieren wollen. Dieser Schritt ist nur erforderlich, wenn Sie mehrere Versionen der Optionssoftware M7-SYS auf dem PG installiert haben. Sie können nur die Versionen auswählen, die für die Baugruppe freigegeben sind (siehe auch Produktinformation M7–SYS).

Im Zielsystem:
Wenn eine Online-Verbindung zum Zielsystem möglich ist, wird in diesem Register die aktuelle Betriebssystemversion im Zielsystem angezeigt, sofern sie ermittelt werden kann.

Betriebssystem auswählen

Selektieren Sie eine Betriebssystem-Konfiguration aus dem Feld ”Mögliche Konfigurationen”. Die Auswahl der Betriebssystem-Konfiguration hängt von der Art der Anwendungsprogramme ab, die auf dem M7-Zielsystem ablaufen sollen. Tabelle 3-1 zeigt Ihnen, wann Sie welches Betriebssystem auswählen. Beachten Sie bitte auch die Hardware-Abhängigkeiten in Tabelle 1-4 auf Seite 1-6.

Tabelle 3-1 Betriebssystem-Konfigurationen

Anwendungsprogramme	Betriebssystem-Konfiguration
nur M7 RMOS32-Applikationen	M7 RMOS32
M7 RMOS32- und MS-DOS-Applikationen	M7 RMOS32 & MS-DOS

Der Speicherbedarf von M7 RMOS32 ohne MS-DOS beträgt maximal 3 MB auf dem Zielmedium. Hinzu müssen Sie noch den Speicherbedarf Ihrer Anwendungsprogramme addieren.

Welche Betriebssystemkonfigurationen auf welchen Massenspeichern freigegeben sind entnehmen Sie der Produktinformation zu M7-SYS. Nichtfreigegebene Konfigurationen können nicht installiert werden.

Medium auswählen

Im Auswahlfeld ”Medium” sind folgende **Installationsmedien** angeboten:

1. MPI/RFS:

   Wählen Sie ”MPI/RFS” (RFS = **R**emote **F**ile **S**ystem), wenn das Betriebssystem auf die Festplatte des M7-Zielsystems installiert werden soll. Um dieses Installationsmedium benutzen zu können, muß eine MPI-Verbindung zwischen dem Erstellsystem und dem Zielsystem aufgebaut werden.

   Das Betriebssystem wird über MPI/RFS immer auf die Festplatte des M7-Zielsystems installiert.

   Bei der Installation über MPI/RFS benötigen Sie immer auch ein **Bootmedium** (siehe Seite 3-5).

2. Memory Card

   Wählen Sie ”Memory Card”, wenn das Betriebssystem auf die Memory Card installiert werden soll. Das Betriebssystem und die Anwendungsprogramme werden vom PG auf die Memory Card übertragen.

   Anschließend wird die Memory Card in das M7-Zielsystem eingelegt und das Zielsystem davon gebootet.

   Um eine Memory Card zu benutzen, benötigen Sie ein PG 720/740/760 oder einen PC mit externem Prommer.

Hinweis

Eine 1,44 MB Diskette kann ein Minimalsystem M7 RMOS32 enthalten, kann jedoch nicht als Zielmedium für die Installation des Betriebssystems auf M7-300/400 verwendet werden. Eine Diskette können Sie als Bootmedium oder als Datenträger für Anwendungsprogramme benutzen.

Bei der Auswahl des Mediums werden Sie durch eine passende Auswahlliste unterstützt. Z.B., wird ”Memory Card” nur als Medium angeboten, wenn Ihr PC/PG auch ein Laufwerk für Memory Cards besitzt.

Lokales Laufwerk und Partner Laufwerk auswählen

Wenn Sie das Medium ”MPI/RFS” für die Installation benutzen, wird eine MPI–Verbindung zwischen dem lokalen Laufwerk des PC/PG und einem Laufwerk auf dem Zielsystem aufgebaut.

Lokales Laufwerk:
In der Auswahlliste werden Ihnen die freien Laufwerksbezeichnungen auf dem PC/PG angezeigt, von denen Sie ein beliebiges Laufwerk auswählen können.

Das lokale Laufwerk muß nur auf PC/PG mit Windows 95 ausgewählt werden, unter Windows NT kann dieses Feld nicht benutzt werden.

Partner Laufwerk:
In der Auswahlliste werden Ihnen die Laufwerksbezeichnungen auf dem M7-Zielsystem angezeigt, aus denen Sie den gewünschten Massenspeicher auswählen können. Üblicherweise (falls nicht anders eingestellt) sind die Laufwerke wie folgt zugeordnet:

Tabelle 3-2 Zuordnung der Laufwerke (Voreinstellung) auf M7-Zielsystem

Laufwerk	Partnerlaufwerk-Bezeichnung	
	MS-DOS	M7 RMOS32
Memory Card	A: oder B:	M0:
Festplatte	C:, D:, ...	C:, D:, ...
Onboard Silicon Disk	D:, E:, ... mit Festplatte C: ohne Festplatte	M1:

Bootmedium

Wenn Sie das Betriebssystem auf der Festplatte des M7-Zielsystems installieren, benötigen Sie zusätzlich ein Bootmedium. Als Bootmedium bezeichnen wir einen Datenträger, von dem aus das Zielsystem nach Einschalten der Versorgungsspannung bootet. Das Bootmedium enthält ein minimales M7 RMOS32 Betriebssystem. Beim Booten werden die Teile des Betriebssystems, die für den Ablauf der Anwendungsprogramme und für die Kommunikation notwendig sind, in den Arbeitsspeicher geladen.

Nach dem Hochlauf vom Bootmedium kann eine MPI-Verbindung zwischen PC/PG und M7-Zielsystem aufgebaut werden.

Bootfähige Datenträger für M7-Zielsysteme sind:

- 3,5”/1,44 MB-Disketten oder
- Memory Cards $\geq$ 2 MB

MS-DOS installieren

Bevor Sie die Betriebssystem-Konfiguration mit MS-DOS installieren, müssen Sie MS-DOS V6.22 auf dem M7-Zielsystem installieren.

Festplatte partitionieren

Wenn Sie das Betriebssystem auf die Festplatte installieren, empfehlen wir aus Gründen der Datensicherheit bei Netzausfall (siehe Kapitel 3.2), zwei Partitionen anzulegen. Die Festplatte können Sie mit folgenden Aufrufen partitionieren:

- **hdpart** unter M7 RMOS32 (siehe Kapitel 5)
- **fdisk** unter MS-DOS

Zielmedium formatieren

In der Regel wird das Zielmedium vor der Erstinstallation des Betriebssystems formatiert. Bei den M7-Betriebssystem-Konfigurationen müssen Sie das Zielmedium in folgenden Fällen formatieren:

Betriebssystem	Zielmedium wird formatiert ...
M7 RMOS32	vor jeder Neuinstallation oder Nachinstallation, da M7 RMOS32, wenn es ohne MS-DOS läuft, immer an den Anfang des Speichers geschrieben werden muß.
M7 RMOS32 mit MS-DOS	vor der ersten Installation von MS-DOS.

Während der Installation von M7 RMOS32 ohne MS-DOS werden Sie aufgefordert, das Zielmedium Festplatte zu formatieren. Wie Sie dabei vorgehen, ist im Kapitel 5 beschrieben.

Installieren in ein MPI-Subnetz

Wenn Sie M7 RMOS32 auf einer Baugruppe installieren, deren MPI-Parameter von den Default-Einstellungen abweichen, wird beim Booten ein schwerwiegender Fehler angezeigt und die Installation wird abgebrochen.

Abhilfe:
Um die Installation durchzuführen, haben Sie folgende Möglichkeiten:

- Trennen Sie die Baugruppe vom MPI-Subnetz und schließen Sie das PG lokal an.
- Konfigurieren Sie das MPI-Subnetz vorerst mit Default-Werten, führen Sie die Installation durch und konfigurieren Sie anschließend das MPI-Subnetz mit den gewünschten Werten.

3.2 Datensicherheit bei Netzausfall

Konzept

Der Automatisierungsrechner M7-300/400 verfügt über mehrere unterschiedliche Massenspeicher: Festplatte, Diskette, Memory Card und OSD, deren Dateisysteme vom Betriebssystem verwaltet werden.

Vorsicht

Tritt ein Netzausfall auf, während auf einen Massenspeicher geschrieben wird, so können Daten verfälscht werden oder verlorengehen.

Da die Systemsoftware (Betriebssystem, Konfigurationsdateien, etc) auch auf einem Massenspeicher liegt, kann ein Netzausfall während eines Schreibzugriffs dazu führen, daß das System nicht mehr gestartet werden kann.

Zur Lösung dieses Problems empfehlen wir, grundsätzlich mit mindestens zwei Massenspeichern (oder zwei Partitionen auf der Festplatte) zu arbeiten:

- einem Bootlaufwerk (Bootpartition), das das Betriebssystem und die systemrelevanten Dateien enthält und auf das keine schreibenden Zugriffe im laufenden Betrieb stattfinden und
- einem Datenlaufwerk (Datenpartition), das die Anwendungsprogramme und die Bereiche der Festwert-, Backup- und Ladespeicher enthält und auf welchem schreibende Zugriffe im laufenden Betrieb erlaubt sind.

Auf der Festplatte ist das Bootlaufwerk immer C:.

Prinzipielles Vorgehen

Um die Konsistenz der Daten auf dem Massenspeicher im Falle eines Netzausfalles sicherzustellen, haben Sie folgende Möglichkeit:

- Installieren Sie das Betriebssystem auf das Bootlaufwerk, d.h. auf eine eigene Partition der Festplatte oder auf einen eigenen Massenspeicher. Sorgen Sie dafür, daß auf der Partition oder den Massenspeicher des Betriebssystems keine schreibenden Zugriffe im laufenden Betrieb stattfinden. Dadurch stellen Sie sicher, daß das Betriebssystem und die Systemdaten nach einem Netzausfall intakt sind, d.h. daß ein Neustart immer durchgeführt werden kann.
- Legen Sie die Verzeichnisse für den Backup-Speicher, den permanenten Ladespeicher und den Festwertspeicher nicht auf das gleiche Laufwerk wie das Betriebssystem sondern auf das Datenlaufwerk, auf welches Sie im laufenden Betrieb schreiben. Dafür müssen Sie die Umgebungsvariablen BACKDIR, RAMDIR und ROMDIR in der Datei \ETC\INITTAB auf dem Bootlaufwerk mit den entsprechenden Pfadnamen versehen.
- Installieren Sie die Anwendungsprogramme nicht auf das gleiche Laufwerk wie das Betriebssystem, sondern auf das Datenlaufwerk.

3.3 Installieren von M7 RMOS32 auf Memory Card

Ausgangszustand

Ihr M7-Zielsystem hat kein Festplatten- und Disketten-Laufwerk.

Voraussetzung

In diesem Fall können Sie die Memory Card als Zielmedium benutzen. Auf einer Memory Card findet ein vollständiges M7 RMOS32 Betriebssystem mit Anwendungsprogrammen Platz.

Sie benötigen:

- ein Memory-Card-Laufwerk an Ihrem PG 720/740/760 oder einen PC mit externem Prommer
- eine Memory Card $\geq$ 4 MB

Vorgehen

Um ein M7 RMOS32 Betriebssystem auf einer Memory Card in Betrieb zu setzen, sind folgende Schritte erforderlich:

1. Selektieren Sie in Ihrem Projekt das M7-Programm, das der M7-CPU/FM zugeordnet ist.
2. Starten Sie die M7-Verwaltung mit dem Menübefehl:
   Zielsystem ▸ M7-Zielsystem verwalten
3. Öffnen Sie das Register ”BS installieren”
4. Installieren Sie ein M7 RMOS32 Betriebssystem lokal auf der Memory Card indem Sie folgende Auswahlen treffen:
   – Medium: ”Memory Card”
   – Mögliche Konfiguration: ”M7 RMOS32”
5. Wählen Sie die Schaltfläche ”Installieren”. Im Dialogfeld erhalten Sie Meldungen über die aktuellen Vorgänge.

   Ergebnis: Das Betriebssystem und die vollständige M7-Systemsoftware werden auf die Memory Card übertragen.
6. Übertragen Sie Ihr Anwendungsprogramm mit allen zugehörigen Projektdaten lokal auf die Memory Card. Schalten Sie dafür in das Register ”Programme” und gehen Sie wie unter ”M7-Programme über Datenträger übertragen” Seite 4-14 vor. Dieser Schritt ist optional.
7. Legen Sie die Memory Card in das M7-Zielsystem ein und starten Sie es erneut. Stellen Sie bei Bedarf das BIOS-Setup ein (siehe **/80/** , **/101/**).

   Ergebnis: Das M7-Zielsystem bootet mit dem neuen Betriebssystem. Ihr Anwendungsprogramm wird gestartet.

3.4 Installieren von M7 RMOS32 mit MS-DOS auf Memory Card

Ausgangszustand

Ihr M7-Zielsystem hat kein Festplatten- und Disketten-Laufwerk.

Voraussetzung

In diesem Fall benutzen Sie die Memory Card als Zielmedium. Auf einer Memory Card findet ein vollständiges Betriebssystem M7 RMOS32 mit Anwendungsprogrammen Platz. Abhängig von der Größe der Memory Card ($\geq$ 4 MB) können Sie darauf zusätzlich auch MS-DOS oder eine Teilmenge davon installieren.

Sie benötigen:

- ein Memory-Card-Laufwerk an Ihrem PG 720/740/760 oder einen PC mit externem Prommer
- eine Memory Card $\geq$ 8 MB
- MS-DOS Installationsdisketten

Prinzipielles Vorgehen

Um ein Betriebssystem M7 RMOS32 mit MS-DOS auf einer Memory Card in Betrieb zu setzen, gehen Sie folgendermaßen vor:

1. Formatieren Sie die Memory Card und installieren Sie MS-DOS oder die Teile davon, die Sie benötigen, auf die Memory Card. Diese Installation führen Sie in der DOS-Box von Windows 95/NT aus.
2. Installieren Sie M7 RMOS32 für MS-DOS auf der Memory Card und starten Sie damit Ihr M7-Zielsystem. Diese Installation führen Sie unter STEP 7 aus.

MS-DOS auf Memory Card installieren

Um MS-DOS auf einer Memory Card zu installieren, sind folgende Schritte erforderlich:

1. Legen Sie die MS-DOS V6.22 Installationsdiskette 1 und die Memory Card in die Laufwerke des PC/PG ein.
2. Rufen Sie auf dem PC/PG in der DOS-Box folgenden Befehl auf:
   unter Windows 95 : <STEP7_Verzeichnis>\s7bin\s7oformx *<m:>* **/s***<a:>*
   unter Windows NT : <STEP7_Verzeichnis>\s7bin\s7ofornx *<m:>* **/s***<a:>*
   wobei:
   - *<m:>* ist die Bezeichnung des Memory Card-Laufwerks auf PC/PG
   - *<a:>* ist die Bezeichnung des Disketten-Laufwerks auf PC/PG

   Ergebnis: Damit wird die Memory Card als Bootmedium formatiert und die Systemdateien von MS-DOS werden darauf kopiert: io.sys, msdos.sys und command.com.

Anschließend müssen Sie die Teile von MS-DOS, die Sie für Ihre Anwendung benötigen, von den MS-DOS-Disketten auf die Memory Card kopieren. Die Daten auf den Installationsdisketten von MS-DOS sind komprimiert und können erst nach dem Dekomprimieren durch Setup verwendet werden. Um Dateien direkt von einer Installationsdiskette zu kopieren, ohne Setup zu verwenden, sind noch folgende Schritte notwendig:

3. Kopieren Sie die Datei EXPAND.EXE von der Installationsdiskette 1 auf die Festplatte des PG/PC.
4. Legen Sie in Laufwerk A: die Diskette ein, die die zu dekomprimierende Datei enthält.

   Wenn Sie nicht wissen, auf welcher Diskette sich die gewünschte Datei befindet, öffnen Sie die Datei PACKING.LST auf der Installationsdiskette 1. Die Datei PACKING.LST enthält der Namen der Dateien, die sich auf den Installationsdisketten befinden.
5. Geben Sie an der Eingabeaufforderung folgendes ein:

   expand *x:\Dateiname1 y:\Verzeichnis\Dateiname2*

   Geben Sie für den Parameter *x* den Kennbuchstaben des Diskettenlaufwerks ein, von dem Sie kopieren. Geben Sie für *Dateiname1* den Namen der komprimierten Datei ein. Geben Sie anstelle des Parameters *y* den Kennbuchstaben des Ziellaufwerks (Memory Card) ein. Für Verzeichnis geben Sie den Namen des Verzeichnisses ein, in das die dekomprimierte Datei geordnet werden soll. Für *Dateiname2* geben Sie den Namen der Datei an, den die Datei nach dem Dekomprimieren erhalten soll.

   Ergebnis: Die komprimierte Datei wird dekomprimiert auf Ihr Ziellaufwerk kopiert.

   Wiederholen Sie diesen Aufruf für alle Dateien, die Sie auf die Memory Card übertragen werden sollen.

Nach Abschluß der Installation von MS-DOS können Sie M7 RMOS32 für MS-DOS auf die Memory Card installieren.

M7 RMOS32 für MS-DOS installieren

Um ein Betriebssystem M7 RMOS32 mit MS-DOS auf einer Memory Card in Betrieb zu setzen, benötigen Sie die Memory Card, auf der Sie MS-DOS installiert haben (siehe Seite 3-10). Folgende Schritte sind erforderlich:

1. Selektieren Sie in Ihrem Projekt das M7-Programm, das der M7-CPU/FM zugeordnet ist.
2. Starten Sie die M7-Verwaltung mit dem Menübefehl:
   Zielsystem ▸ M7-Zielsystem verwalten
3. Öffnen Sie das Register ”BS installieren”
4. Installieren Sie M7 RMOS32 für MS-DOS lokal auf der Memory Card indem Sie folgende Auswahlen treffen:
   – Medium: ”Memory Card”
   – Mögliche Konfiguration: ”M7 RMOS32 & MS-DOS”
5. Wählen Sie die Schaltfläche ”Installieren”. Im Dialogfeld erhalten Sie Meldungen über die aktuellen Vorgänge.

   Ergebnis: Das Betriebssystem und die vollständige M7-Systemsoftware werden auf die Memory Card übertragen.
6. Übertragen Sie Ihr Anwendungsprogramm mit allen zugehörigen Projektdaten lokal auf die Memory Card. Schalten Sie dafür in das Register ”Programme” und gehen Sie wie unter ”M7-Programme über Datenträger übertragen” Seite 4-14 vor. Dieser Schritt ist optional.
7. Legen Sie die Memory Card in das M7-Zielsystem ein und starten Sie Ihr M7-Betriebssystem erneut.

   Ergebnis: Das M7-Zielsystem bootet mit dem neuen Betriebssystem von der Memory Card. Ihr Anwendungsprogramm wird gestartet.

3.5 Installieren von M7 RMOS32 auf Festplatte

Ausgangszustand

Ursprünglich ist auf dem M7-Zielsystem noch kein lauffähiges Betriebssystem installiert und noch keine MPI-Verbindung möglich.

Voraussetzung

Um M7 RMOS32 auf der Festplatte des M7-Zielsystems zu installieren benötigen Sie:

- eine Massenspeicherbaugruppe MSM 378/478 an Ihrem M7-Zielsystem
- ein Bootmedium (Diskette 1,44 MB oder Memory Card $\geq$ 2 MB)

Vorgehen

Folgende Schritte sind erforderlich:

1. Selektieren Sie in Ihrem Projekt das M7-Programm, das der M7-CPU/FM zugeordnet ist und starten Sie die M7-Verwaltung mit dem Menübefehl: **Zielsystem ▸ M7-Zielsystem verwalten.**
2. Öffnen Sie das Register ”BS installieren” und treffen Sie folgende Auswahlen:
   - Medium: ”MPI/RFS”
   - Mögliche Konfiguration: ”M7 RMOS32”
   - Lokales Laufwerk: das erste freie Laufwerk, z.B. F:
   - Partner Laufwerk: C: für Festplatte
3. Wählen Sie die Schaltfläche ”Installieren”.

   Danach werden Sie über Meldungen in Dialogfeldern über das aktuelle Geschehen informiert und zu weiterem Vorgehen angeleitet. Hauptsächlich müssen Sie folgendes tun:
4. Ein Bootmedium (Floppy Disk oder Memory Card) auswählen.
   Ergebnis: Ein minimales M7 RMOS32-Betriebssystem wird auf dem ausgewählten Bootmedium installiert.
5. Das Bootmedium in das Laufwerk des M7-Zielsystems einlegen und das M7-Zielsystem starten. **Ergebnis:** Das M7-Zielsystem bootet mit dem neuen Betriebssystem und eine MPI-Verbindung wird zwischen dem PC/PG und M7-Zielsystem aufgebaut.
6. Über RTI (Remote Terminal Interface) oder an der lokalen Konsole des M7-Zielsystems die Festplatte partitionieren (siehe Kapitel 3.2) und formatieren (siehe auch Kapitel 5.2)

   Ergebnis: Die Festplatte wird partitioniert und formatiert. Danach wird das M7 RMOS32-Betriebssystem und eventuelle Anwendungsprogramme auf die Festplatte des M7-300/400 über die MPI-Verbindung installiert.

   Um Ihr Anwendungsprogramm auf das M7-Zielsystem zu übertragen, öffnen Sie das Register ”Programme” und gehen Sie wie unter ”M7-Programme über MPI/RFS übertragen”, Seite 4-13 vor.
7. Das M7-Zielsystem über Betriebsartenschalter erneut starten und bei Bedarf BIOS-Setup einstellen. **Ergebnis:** Das M7-Zielsystem bootet mit dem neuen Betriebssystem von der Festplatte. Ihr Anwendungsprogramm -falls vorhanden - wird gestartet.

3.6 Installieren von M7 RMOS32 mit MS-DOS auf Festplatte

Ausgangszustand

Ursprünglich ist auf dem M7-Zielsystem noch kein lauffähiges Betriebssystem installiert und noch keine MPI-Verbindung möglich.

Voraussetzung

Um M7 RMOS32 mit MS-DOS auf der Festplatte des M7-Zielsystems zu installieren benötigen Sie:

- eine Massenspeicherbaugruppe MSM 378/478 an Ihrem M7-Zielsystem
- ein Bootmedium (Diskette 1,44 MB oder Memory Card $\geq$ 2 MB)
- MS-DOS Installationsdisketten. **MS-DOS V6.22 muß auf der Festplatte des M7-Zielsystems installiert sein.**

Vorgehen

Um ein Betriebssystem M7 RMOS32 mit MS-DOS- auf einem M7-Zielsystem mit Festplatte zu installieren, sind folgende Schritte erforderlich:

1. Selektieren Sie in Ihrem Projekt das M7-Programm, das der M7-CPU/FM zugeordnet ist und starten Sie die M7-Verwaltung mit dem Menübefehl:
   Zielsystem ▸ M7-Zielsystem verwalten
2. Öffnen Sie das Register ”BS installieren”und treffen Sie folgende Auswahlen:
   – Medium: ”MPI/RFS”
   – Mögliche Konfiguration: ”M7 RMOS32 & MS-DOS”
   – Lokales Laufwerk: das erste freie Laufwerk, z.B. F:
   – Partner Laufwerk: C: für Festplatte
3. Wählen Sie die Schaltfläche ”Installieren”.

   Danach werden Sie über Meldungen in Dialogfeldern über das aktuelle Geschehen informiert und zu weiterem Vorgehen angeleitet. Hauptsächlich müssen Sie folgendes tun:

4. Ein Bootmedium (Floppy Disk oder Memory Card) auswählen. **Ergebnis:** Ein minimales M7 RMOS32 wird auf dem Bootmedium installiert.
5. Laufwerke für Betriebssystem und Daten auswählen (siehe Kapitel 3.2).
6. Das Bootmedium in das Laufwerk des M7-Zielsystems einlegen und das M7-Zielsystem starten. **Ergebnis:** Das M7-Zielsystem bootet mit dem neuen Betriebssystem und eine MPI-Verbindung zwischen dem PC/PG und M7-Zielsystem wird aufgebaut. Danach wird M7 RMOS32 für MS-DOS und eventuelle Anwendungsprogramme auf die Festplatte des M7-300/400 über die MPI-Verbindung installiert.

   Um Ihr Anwendungsprogramm auf das M7-Zielsystem zu übertragen, öffnen Sie das Register ”Programme” und gehen Sie wie unter ”M7-Programme über MPI/RFS übertragen”, Seite 4-13 vor.
7. Das M7-Zielsystem über Betriebsartenschalter erneut starten und bei Bedarf BIOS-Setup einstellen. **Ergebnis:** Das M7-Zielsystem bootet mit dem neuen Betriebssystem von der Festplatte. Ihr Anwendungsprogramm -falls vorhanden - wird gestartet.

3.7 Nachinstallation des M7-Betriebssystems

Ausgangszustand

Ist auf der Festplatte des M7-Zielsystem bereits ein Betriebssystem vorhanden, so können Sie eine **Nachinstallation** über ”MPI/RFS” durchführen, d. h. das Betriebssystem auf Ihrem M7-Zielsystem mit einer neuen Ausgabe überschreiben.

Welche Betriebssystemkonfigurationen auf welchen Massenspeichern freigegeben sind entnehmen Sie der Produktinformation zu M7-SYS.

Hinweis

Bei einer Nachinstallation werden auch die Konfigurationsdateien, z.B. INITTAB und RMOS.INI auf das Zielsystem übertragen. Wenn Sie Änderungen in den Konfigurationsdateien vorgenommen haben, ohne den Menübefehl **Zielsystem ▸ M7-Zielsystem verwalten** zu verwenden, müssen Sie diese vor einer Nachinstallation sichern, da sie ansonsten überschrieben werden.

Nachinstallation auf Festplatte

Die Tabelle 3-3 zeigt Ihnen, was Sie in den unterschiedlichen Fällen der Nachinstallation auf Festplatte tun müssen. Das grundsätzliche Vorgehen ist wie bei der Neuinstallation und ist bereits in den Kapiteln 3.5 und 3.6 beschrieben.

Tabelle 3-3 Nachinstallation auf Festplatte

Vorhanden	Wenn Sie nachinstallieren	
	M7 RMOS32	M7 RMOS32 mit MS-DOS
M7 RMOS32	Wie M7 RMOS32 Neuinstallation	Zielmedium formatieren MS-DOS lokal installieren, und M7 RMOS32 neu installieren
M7 RMOS32 mit MS-DOS	Wie M7 RMOS32 Neuinstallation	Es wird nur eine neue M7 RMOS32-Komponente nachinstalliert

Nachinstallation auf Memory Card

Eine Nachinstallation von M7 RMOS32 auf Memory Card ist immer eine Neuinstallation. Bei M7 RMOS32 mit MS-DOS wird in der Regel nur die M7 RMOS32-Komponente nachinstalliert.

Weitere Information

Ausführliche Beschreibungen der einzelnen Schritte sowie zusätzliche Information zur Installation der Betriebssystem-Konfigurationen finden Sie in der Online-Hilfe der M7-Zielsystemverwaltung.

3.8 Aktualisieren der Firmware

Einführung

Auf den M7-300/400 Zentralbaugruppen und Applikationsbaugruppen befindet sich baugruppenspezifische Firmware, wie z.B. das BIOS. Die Aktualisierung der Firmware können Sie über die Zielsystemverwaltung durchführen.

Die Firmware kann unabhängig oder zusammen mit der Systemsoftware für M7-300/400 aktualisiert werden.

Im Register “FW aktualisieren“ wird die Version der Firmware im Zielsystem und im PG angezeigt. Damit wird die Überprüfung, ob eine Aktualisierung der Firmware notwendig ist, einfacher: Eine Aktualisierung der Firmware ist nur notwendig, wenn die Firmware-Version im Zielsystem älter ist als die Firmware-Version im PG.

Wenn Sie das Betriebssystem über MPI/RFS installieren, wird die Firmware-Version automatisch überprüft. Falls die auf dem M7-Zielsystem vorhandene Firmware-Version nicht zu dem installierten Betriebssystem paßt, erhalten Sie eine entsprechende Meldung.

Voraussetzung

Um die Firmware einer M7-300/400 CPU oder FM zu aktualisieren benötigen Sie ein Bootmedium (Diskette oder Memory Card).

Die Aktualisierung muß im Kontext eines Projektes, das die M7-Stationen (CPUs oder FMs) enthält, mit angewähltem ”M7-Programm” durchgeführt werden.

Vorsicht

Hardwareschaden:
Während des Aktualisierungsvorgangs darf keinesfalls die Netzspannung ausgeschaltet werden, da dadurch die Baugruppe beschädigt werden kann.

Vorsicht

Datenverlust:
Nach der Firmware-Aktualisierung müssen Sie das BIOS-Setup aufrufen (siehe Kapitel 12.4, Referenzhandbuch, Automatisierungssystem S7-400, M7-400, Baugruppendaten bzw. Kapitel 10 im Handbuch Automatisierungssystem M7-300 Aufbauen, CPU-Daten). Wenn Sie mit den Default-Werten arbeiten wollen, speichern Sie diese bitte ab. Haben Sie vor der Aktualisierung Einstellungen im BIOS-Setup geändert, sind diese verloren und müssen neu eingetragen werden.

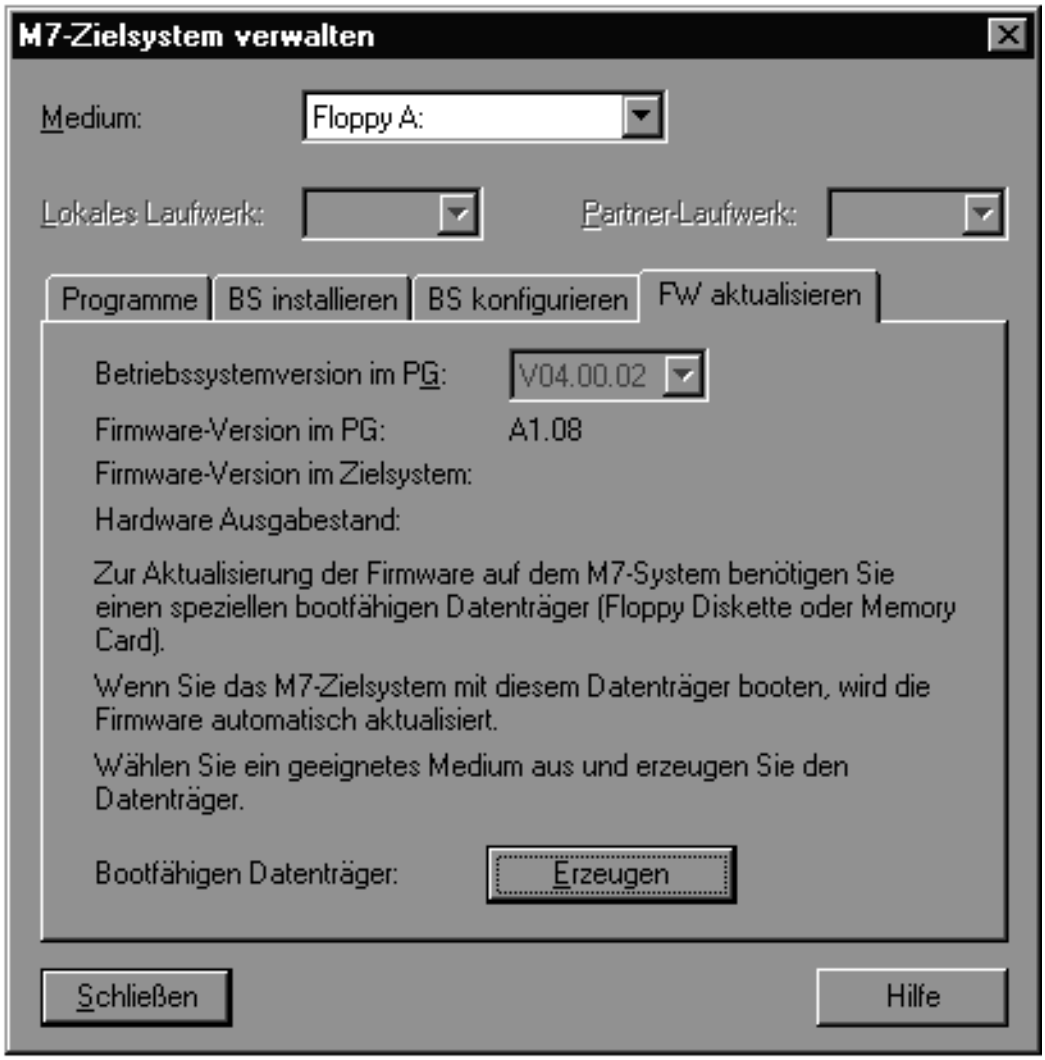

Bild 3-2 Register ”FW aktualisieren”

Versionen und Ausgabestände

Listenfeld “Betriebssystemversion im PG”:

Wählen Sie hier die Version des Betriebssystems, das Sie auf Ihrem M7-Zielsystem installieren wollen oder bereits installiert haben. Dieser Schritt ist nur erforderlich, wenn Sie mehrere Versionen der Optionssoftware M7-SYS auf dem PG installiert haben.

Firmware-Version, Hardware-Ausgabestand:

Folgende Versionen/Ausgabestände werden angezeigt, wenn eine MPI-Verbindung zum M7-Zielsystem besteht:

- aktuelle Firmware-Version im PG
- im Zielsystem vorhandene Firmware-Version,
- Hardware-Ausgabestand des Zielsystems

Vorgehen

Um die Firmware eines M7-Zielsystems zu aktualisieren, gehen Sie wie folgt vor:

1. Selektieren Sie den M7-Programmbehälter, welcher der M7-Baugruppe (CPU oder FM) zugeordnet ist.
2. Rufen Sie den Menübefehl
   Zielsystem ► M7-Zielsystem verwalten auf und öffnen Sie das Register ”FW aktualisieren” (siehe Bild 3-2).
3. Wählen Sie das Bootmedium, Floppy oder Memory Card.
4. Wählen Sie die Version des Betriebssystems aus dem Listenfeld “Betriebssystemversion im PG” aus. Dieser Schritt ist nur erforderlich, wenn Sie mehrere Versionen der Optionssoftware M7-SYS auf dem PG installiert haben.
5. Betätigen Sie die Schaltfläche ”Erzeugen”.

   Ergebnis: Das Bootmedium wird (mit einer Vorwarnung) formatiert und die neue Firmware wird darauf installiert.
6. Legen Sie das Bootmedium in das M7-Zielsystem ein und starten Sie die M7-CPU/FM.

   Ergebnis: Wenn Sie das M7-Zielsystem mit diesem Datenträger booten, wird die Firmware automatisch aktualisiert. Die Aktualisierung ist abgeschlossen, wenn folgende LED konstant leuchtet:

   – die USR-LED bei M7-300 CPU/FM

   – die USR1-LED bei M7-400 CPU/FM
7. Entfernen Sie das Bootmedium aus dem M7-Zielsystem und booten Sie es von dem voreingestellten Massenspeicher (Memory Card, Festplatte, etc)

Hinweis

Wenn die Firmware auf dem Bootmedium inkompatibel zum Baugruppentyp oder älter als die bereits vorhandene Firmware-Version ist, wird die Aktualisierung **nicht durchgeführt** und die Fehler-LED (SF-LED bei M7-300 und INTF-LED bei M7-400) leuchtet auf. Dies gilt auch wenn versucht wird, die gleiche Firmware-Version zu installieren.

Fehlerfall

Im Fehlerfall tun Sie folgendes:

1. Entfernen Sie das Bootmedium aus dem M7-Zielsystem.
2. Überprüfen Sie ggf. ob die Version des BIOS der M7-300/400 Baugruppe höher als die Version der neuen Firmware ist. In diesem Fall ist eine Aktualisierung nicht erforderlich und auch nicht möglich.
3. Überprüfen Sie, ob die Baugruppe in Ihrem Projekt mit dem Baugruppentyp des M7-Zielsystems übereinstimmt. Ist dies nicht der Fall, passen Sie Ihr Projekt entsprechend an, und führen Sie die Aktualisierung der Firmware erneut durch.

3.9 Ändern der Konfigurationsdateien

Konfigurationsdateien

M7-SYS RT V 4.0 unterstützt die Änderung der Konfigurationsdateien von M7 RMOS32 im SIMATIC Manager von STEP 7. Im Register ”BS konfigurieren” des Dialogfelds “M7-Zielsystem verwalten” können Sie folgende Konfigurationsdateien des Betriebssystems auswählen und bearbeiten.

Datei	Beschreibung	Siehe
RMOS.INI	Konfigurationsdatei von M7 RMOS 32	Kap. 3.10
INITTAB	Initialisierungsdatei der M7-Systemserver	Kap. 3.11
CLISTART.BAT	Initialisierungsdatei des CLI	Kap. 4.2
HOSTS	Teilnehmerliste für TCP/IP	/280/, Kap. 9
SERVICES	Dienste von TCP/IP	/280/, Kap. 9

Die Dateien werden baugruppenspezifisch im jeweiligen “M7-Programm”–Behälter des Projektes abgelegt.

Prinzipielles Vorgehen

Sie können die Konfigurationsdateien folgendermaßen ändern:

- Die auf dem PC/PG im Projekt vorhandenen Dateien ändern und anschließend auf das M7-Zielsystem laden. Diese Vorgehensweise ist insbesondere bei der Installation einer neuen Version des Betriebssystems aber auch allgemein zu empfehlen.
- Die auf dem M7–Zielsystem vorhandenen Konfigurationsdateien ins PG laden, ändern und wieder auf das M7-Zielsystem laden. Diese Vorgehensweise ist für Änderungen einer bereits installierten Betriebssystemversion geeignet, z.B. Eintragen neuer Kommunikationsteilnehmer in die HOSTS-Datei.

Geänderte Konfigurationsdateien können Sie in das M7-Zielsystem laden:

- einzeln, mit der Schaltfläche “Laden” im Register “BS konfigurieren” oder
- zusammen mit dem gesamten Betriebssystem bei der Installation im Register “BS installieren”.

Wenn Sie die Default-Konfigurationsdateien benutzen möchten, sind keine Änderungen erforderlich.

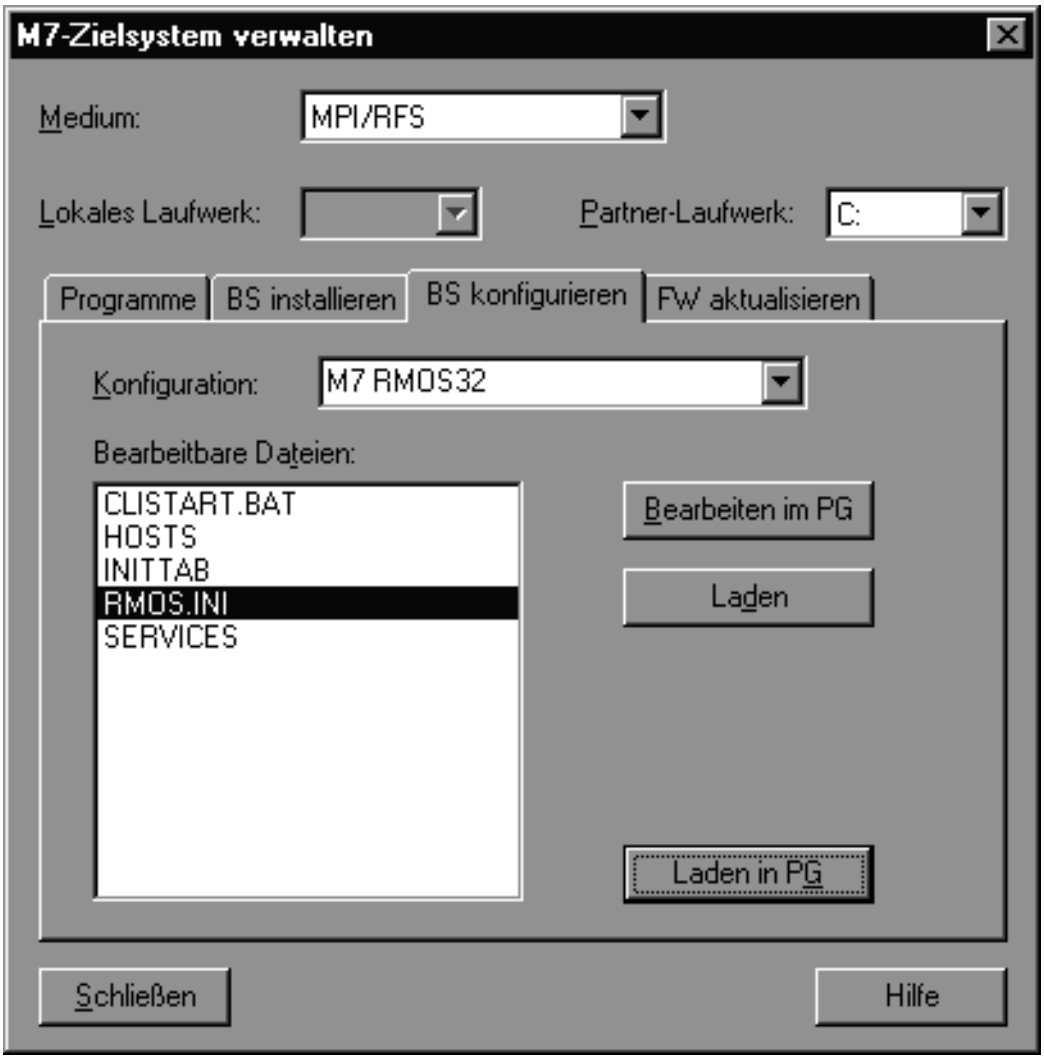

Bild 3-3 Register “BS konfigurieren”

Hinweis

Es können nur die Konfigurationsdateien der Systemsoftware M7-SYS RT V4.0 geändert und in den M7-300/400 geladen werden. Für ältere Versionen wird dies nicht unterstützt. Wir empfehlen Ihnen, soweit wie möglich alle Änderungen der Konfigurationsdateien nur im PC/PG durchzuführen und die geänderten Dateien anschließend in den M7-300/400 zu laden. Dadurch vermeiden Sie Inkonsistenzen zwischen dem Projekt im PC/PG und dem Betriebssystem auf dem M7-300/400.

Ändern der im Projekt vorhandenen Konfigurationsdateien

Gehen Sie wie folgt vor:

1. Selektieren Sie den M7-Programmbehälter, welcher der M7-Baugruppe (CPU oder FM) zugeordnet ist.
2. Rufen Sie den Menübefehl **Zielsystem ▸ M7-Zielsystem verwalten** auf und öffnen Sie das Register ”BS konfigurieren” (siehe Bild 3-2).
3. Wählen Sie im Feld “Medium” das Übertragungsmedium für das Laden

   Wenn Ihr Zielsystem über eine Festplatte verfügt, wählen Sie hier ”MPI/RFS”. Ansonsten wählen Sie einen anderen Datenträger.
4. Wählen Sie das lokale Laufwerk und das Partner–Laufwerk. Diese Optionen können Sie nur angeben, wenn Sie als Medium ”MPI/RFS” eingestellt haben.

   Lokales Laufwerk (nur unter Windows 95):
   Wählen Sie den Laufwerksbuchstaben, über den Sie das Zielsystem ansprechen wollen. Vorzugsweise sollten Sie den ersten verfügbaren Buchstaben verwenden.

   Partner–Laufwerk:
   Wählen Sie das Laufwerk, das Sie ansprechen wollen, d.h. das Laufwerk auf dem auch das Betriebssystem installiert ist
5. Wählen Sie im Feld “Konfiguration” das Betriebssystem aus, dessen Konfigurationsdateien Sie bearbeiten möchten.
6. Wählen Sie aus der Liste “Bearbeitbare Dateien” die Datei aus, die Sie bearbeiten möchten.
7. Betätigen Sie die Schaltfläche ”Bearbeiten in PG”
   Die ausgewählte Datei wird mit dem Notepad–Editor geöffnet.
8. Bearbeiten Sie die Datei und speichern Sie sie ab.
9. Wiederholen Sie die Schritte 6. bis 8. für jede Datei, die Sie ändern möchten.
10. Laden Sie die Konfigurationsdateien indem Sie:

- die Schaltfläche ”Laden” betätigen, um die markierte(n) Konfigurationsdatei(en) auf das angezeigte Medium zu laden oder
- installieren Sie diese Betriebssystem–Konfiguration auf den M7-300/400. Dabei werden automatisch die geänderten Konfigurationsdateien verwendet.

Ändern der im M7-Zielsystem vorhandenen Konfigurationsdateien

Gehen Sie wie folgt vor:

1. Selektieren Sie den M7-Programmbehälter, welcher der M7-Baugruppe (CPU oder FM) zugeordnet ist.
2. Rufen Sie den Menübefehl **Zielsystem ▸ M7-Zielsystem verwalten** auf und öffnen Sie das Register ”BS konfigurieren” (siehe Bild 3-2).
3. Wählen Sie im Feld “Medium” das Übertragungsmedium für das Laden, bzw. das Laden in PG.

   Wenn Ihr Zielsystem über eine Festplatte verfügt, wählen Sie hier ”MPI/RFS”. Ansonsten wählen Sie einen anderen Datenträger.
4. Wählen Sie das lokale Laufwerk und das Partner–Laufwerk. Diese Optionen können Sie nur angeben, wenn Sie als Medium ”MPI/RFS” eingestellt haben.

   Lokales Laufwerk (nur unter Windows 95):
   Wählen Sie den Laufwerksbuchstaben, über den Sie das Zielsystem ansprechen wollen. Vorzugsweise sollten Sie den ersten verfügbaren Buchstaben verwenden.

   Partner–Laufwerk:
   Wählen Sie das Laufwerk, das Sie ansprechen wollen, d.h. das Laufwerk auf dem auch das Betriebssystem installiert ist
5. Wählen Sie im Feld “Konfiguration” das Betriebssystem aus, dessen Konfigurationsdateien Sie bearbeiten möchten.
6. Wählen Sie aus der Liste “Bearbeitbare Dateien” die Datei aus, die Sie bearbeiten möchten.
7. Betätigen Sie die Schaltfläche ”Laden in PG”
   Die ausgewählte Datei wird vom M7–Zielsystem ins PG geladen und überschreibt die im Projekt vorhandene gleichnamige Datei.
8. Betätigen Sie die Schaltfläche ”Bearbeiten in PG”
   Die ausgewählte Datei wird mit dem Notepad–Editor geöffnet.
9. Bearbeiten Sie die Datei und speichern Sie sie ab.
10. Wiederholen Sie die Schritte 5. bis 8. für jede Datei, die Sie ändern möchten.
11. Betätigen Sie die Schaltfläche ”Laden”, um die markierte(n) Konfigurationsdatei(en) auf das angezeigte Medium zu laden.

Wann werden die Änderungen wirksam?

Die Änderungen der Konfigurationsdateien werden folgendermaßen wirksam:

Datei	**Änderungen werden wirksam..**
RMOS.INI und INITTAB	beim nächsten Neustart des M7-300/400
CLISTART.BAT	beim nächsten Start des CLI (<CTRL>+<R>)
HOSTS und SERVICES	beim nächstem Zugriff über eine der Funktionen der Socket-Bibliothek

3.10 Die Datei RMOS.INI

Übersicht

Die Datei RMOS.INI ist eine Software-Konfigurationsdatei für das Betriebssystem M7 RMOS32. Es handelt sich um eine Datei im ASCII-Textformat. Sie können diese Datei editieren, um bei Bedarf das Verhalten des Betriebssystems an Ihre Erfordernisse anzupassen. Die Datei wird beim Hochlauf des Betriebssystems gelesen und ausgewertet. Neu eingegebene Parameter sind nach Neustart des Betriebssystems wirksam.

Hinweis

Das Betriebssystem ist für Ihre M7-300/400 Hardware-Konfiguration vorkonfiguriert, es bedarf in der Regel keiner Änderungen in der Datei RMOS.INI.

Bitte ändern Sie, wenn erforderlich, nur die nachfolgend beschriebenen Teile der Datei RMOS.INI.

Kommando `run`

Das Kommando `run` erlaubt den Aufruf eines ausführbaren Kommandos aus der Datei RMOS.INI. CLI-Kommandos stehen für `run` nicht zur Verfügung. Im Gegensatz zu den anderen Konfigurationsangaben, die innerhalb der entsprechenden Sektion plaziert werden müssen, kann `run` an jeder beliebigen Stelle in der Datei stehen.

Bei Erreichen der entsprechenden Zeile wird das mit `run` angegebene Kommando geladen und ausgeführt. Nach seiner Beendigung wird es wieder entfernt und die Abarbeitung der Datei wird fortgesetzt.

Beispiel:

Das `run` Kommando, um die deutsche Tastaturbelegung zu aktivieren, lautet:

```
RUN=M0:\BIN\chgkbd KBDGER
```

Struktur der Datei

Durch Schlüsselwörter in eckigen Klammern ist die Datei in Sektionen unterteilt. Jede Sektion kann mehrere Einträge enthalten. Jeder Eintrag ist in einer Zeile definiert. Die Wertzuweisung erfolgt über ein Gleichheitszeichen, gefolgt von einem oder mehreren Parametern. Die Einträge können gelöscht werden, dann werden die Voreinstellungen übernommen. Mit dem Semikolon oder der Zeichenfolge REM können Sie in der Zeile nachfolgende Zeichen auskommentieren.

Sektion RMOS

In der Sektion RMOS können u. A. folgende Eintragungen stehen:

[RMOS]	Kennzeichnung der Sektion RMOS
picenable	Freigabe bestimmter Hardware-Interrupt-Request-Eingänge der programmierbaren Interrupt-Steuerbausteine. Treiber und andere RMOS-Routinen, die auf Hardware-Interrupts aufsetzen, schalten die betreffenden Eingänge selber frei. Hier können Sie zusätzlich Interrupteingänge freischalten, die Sie für eigene Interruptroutinen benötigen. **Beispiel** für Freigabe von IRQ12 und IRQ15: picenable=IRQ12,IRQ15
logsvcx	Hier wird festgelegt, ob Fehlermeldungen der Systemaufrufe an den Bildschirm ausgegeben werden sollen. Mögliche Angaben: YES Bildschirmausgabe NO Keine Bildschirmausgabe **Beispiel** für keine Bildschirmausgabe: logsvcx=NO
sysdev	Definition der Systemkonsole. Es wird der Name des Gerätetreibers angegeben (siehe auch Anhang A.2). Mögliche Angaben: BYT_COM1/2 für COM1 bzw. COM2 BYT_EGA1 bis 3 für EGA/VGA 1 bis 3 (nur M7 RMOS32 ohne MS-DOS) none keine Systemkonsole **Voreinstellung:** sysdev=none **Beispiel** für Systemkonsole auf COM1: sysdev=BYT_COM1
device	Das Kommando `device` lädt einen Treiber oder erzeugt eine Treiber–Unit. Die Syntax und die Parameter sind im Anhang B.9 beschrieben. Die Aufrufe des Kommandos `device` in der Datei RMOS.INI und im CLI unterscheiden sich durch folgendes: Hinter dem Kommando `device` kommt: • ein Leerzeichen – im CLI • ein "="–Zeichen – in der RMOS.INI **Beispiele:** 3964-Treiber laden ohne eine Unit zu erzeugen: DEVICE=\M7RMOS32\3964.DRV Treiber-Unit für SER8250 an COM2 erzeugen: DEVICE=SER8250 COM2 IRQ:3 BASE:0x2F8 MODE:9600–N–8–1

Hinweis

Schalten Sie die Bildschirmausgabe der Fehlermeldungen

logsvcx=YES

nur zu Testzwecken ein. Im laufenden Betrieb kann es zu hohen Taskwechselzeiten in der Ausführung Ihres Programms kommen.

Vorsicht

Wenn Sie auf einer der seriellen Schnittstellen COM1/COM2 eine Systemkonsole eingerichtet haben mit dem Eintrag:

sysdev=BYT_COM1

dürfen Sie diese serielle Schnittstelle nicht für die Kommunikation mit einem der ladbaren Treiber 3964R oder SER8250 verwenden. Ansonsten kann es zu folgendem Verhalten kommen: Die Ausgabe der Systemkonsole wird vom neuen Treiber gesteuert. Sie wird direkt auf den Port umgeleitet und kann zu Konflikten mit dem Kommunikationspartner führen, z.B. wenn eine Fehlermeldung auf der Systemkonsole ausgegeben wird.

Sektion RMOS-DOS

In der Sektion RMOS-DOS können u. A. folgende Eintragungen stehen.

[RMOS-DOS]	Kennzeichnung der Sektion RMOS-DOS
RESERVE_COMx	Dieser Eintrag reserviert die Schnittstelle COM1, COM2, usw. für M7 RMOS32. **Voreinstellung:** REM RESERVE_COM2 Per Voreinstellung ist dieser Eintrag auskommentiert, die COM2-Schnittstelle ist auch für DOS verfügbar. Entfernen Sie die Zeichenfolge REM, um die Schnittstelle COM2 ausschließlich für M7 RMOS32 zu reservieren.
MAX_PORT_ NUMBER	Dieser Eintrag schaltet die Portspiegelung aus. **Voreinstellung:** MAX_PORT_NUMBER = 0XFFFF **Der Wert ist voreingestellt und darf nicht geändert werden.**

3.11 Die Datei INITTAB

Inhalt

Die Datei INITTAB enthält Aufrufe zum Starten der M7–Systemserver und der Anwendungsprogramme im ASCII-Textformat. Die Datei wird beim Hochlauf des Betriebssystems gelesen und ausgewertet. Neu eingegebene Einträge sind nach Neustart des Betriebssystems wirksam.

Struktur der Datei

Ein Eintrag hat folgende Syntax:

<Startkennung> <Pfad> [<Aufrufparameter>]

Beispiel:

0 \SRV\BASICSRV.EXE 340 1

Die Startkennung kann 0, 1 oder 2 sein. Die Werte 0 und 1 sind für die M7-Systemserver reserviert. Anwendungsprogramme werden mit der Startkennung 2 eingetragen.

Der Pfad wird relativ zum Hauptverzeichnis des Laufwerks angegeben.

Aufrufparameter sind optional.

Jeder Eintrag ist in einer Zeile definiert. Mit dem #–Zeichen können Sie in der Zeile nachfolgende Zeichen auskommentieren.

Vorsicht

Die M7-Systemserver sind für Ihre M7-300/400 Hardware- und Software-Konfiguration vorkonfiguriert, diese Aufrufe dürfen nicht geändert werden. Änderungen dieser Aufrufe können das Systemverhalten beeinträchtigen!

Bitte ändern Sie, wenn erforderlich, nur die Aufrufe der Anwenderprogramme.

Bedienen und überwachen

4

In diesem Kapitel

In diesem Kapitel erfahren Sie:

- wie Sie das M7-Zielsystem über Remote Terminal bedienen
- wie Sie mit dem Kommandozeileninterpreter CLI arbeiten
- wie Sie Anwendungsprogramme auf das M7-Zielsystem übertragen und starten
- welche Abfragefunktionen Sie benutzen können

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
4.1	Bedienen über Remote Terminal	4-1
4.2	Arbeiten mit dem Kommandozeileninterpreter CLI	4-4
4.3	Programme auf M7-Zielsystem übertragen und löschen	4-10
4.4	Starten der Anwendungsprogramme auf dem M7-Zielsystem	4-16
4.5	Abfrage- und Steuerfunktionen für M7-300/400	4-17

4.1 Bedienen über Remote Terminal

Was ist ”Remote Terminal” ?

Das Remote Terminal Interface (RTI) ermöglicht es, ein M7-Zielsystem vom PC/PG aus wie von einem lokalen Terminal unter M7 RMOS32 zu bedienen. Für die Kommunikation zwischen M7-Zielsystem und PC/PG ist eine MPI-Verbindung erforderlich.

Über Remote Terminal haben Sie folgende Möglichkeiten:

- Arbeiten mit dem Kommandozeileninterpreter CLI
- Arbeiten mit dem Low-Level Debugger

Remote Terminal starten

Um das Remote Terminal zu starten:

1. Wählen Sie im Programm Manager mit der Schaltfläche ”Start”:
   Start ▸ SIMATIC▸Step 7 ▸ Remote Terminal

   Ein Dialogfeld zur Eingabe der zugehörigen MPI-Adresse wird geöffnet (siehe Bild 4-1.

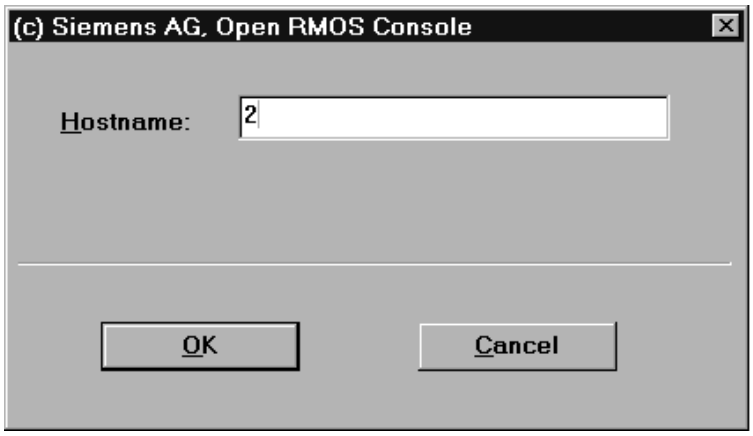

Bild 4-1 Dialog zur Eröffnung des Remote Terminals

2. Geben Sie die zugehörige MPI-Adresse (im Standardfall 2) in das Feld ”Hostname” ein und wählen Sie die Schaltfläche *OK*. Das Dialogfeld wird geschlossen und eine RMOS-Konsole geöffnet.

MPI-Adressen

Bei den Baugruppen FM 456 müssen Sie die Adresse in folgender Form eingeben: ***<H>,<T>*** wobei

- ***<H>*** die MPI-Adresse der CPU 4xx im MPI-Segment und
- ***<T>*** die Kennung (TSAP-ID) der FM 456 ist.

TSAP-ID

Die TSAP-ID ist abhängig von der Rack- und Steckplatznummer der FM456 und setzt sich wie folgt zusammen:

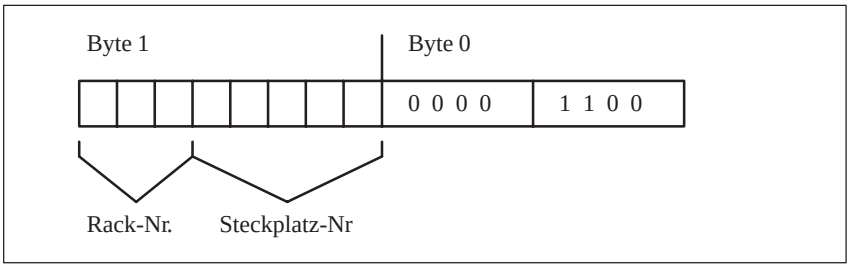

Bild 4-2 Aufbau der TSAP-ID

Beispiel:
Die FM456 in Rack 1 auf Steckplatz 4 (240CH) hat die TSAP-ID: 9228 und die CPU hat die MPI-Adresse 3. In diesem Fall lautet die Eingabe:

3,9228

Arbeiten mit dem CLI

Um im Remote Terminal den Kommandozeileninterpreter CLI zu starten, geben Sie **<CTRL> R** ein. Die Abfrage nach dem Startup-File können Sie mit der Eingabe-Taste beenden. Dadurch wird die voreingestellte Startup-Datei CLISTART.BAT ausgeführt und der CLI gestartet.

Weitere Informationen finden Sie im Kapitel 4.2 und im Anhang B.

Arbeiten mit dem Low-Level Debugger

Um im Remote Terminal den Low-Level Debugger zu starten, geben Sie **<CTRL> D** ein.

Weitere Informationen finden Sie im Kapitel 6 und im Anhang C.

Remote Terminal beenden

Die Remote Terminal Session kann jederzeit durch Eingabe von **<CTRL> Z** beendet werden.

4.2 Arbeiten mit dem Kommandozeileninterpreter CLI

Aufgabe des CLI: Bedienoberfläche

Der Kommandozeileninterpreter CLI (engl. Command Line Interpreter) ist eine Bedienoberfläche für das Betriebssystem M7 RMOS32. Über den CLI können Kommandos und nachladbare Programme gestartet und interaktiv bedient werden.

Funktionsteile

Der CLI besteht aus den folgenden Teilen:

- CLI-Task (Shell)
- Programm-Manager
- Nachladbare Kommandos
- Ablaufumgebung für nachladbare Tasks
- CLI Startup-Batch-Datei CLISTART.BAT

CLI-Task

Der CLI ist im RMOS-System eine dynamische Task. Ein laufender CLI wird als Session bezeichnet. Je Konsole ist eine Session möglich. Die Konsole, an der eine CLI-Session gestartet wurde, ist für diese CLI-Session die Standard-Ein- und Ausgabeeinheit. Die CLI-Task baut auf der Funktionsschnittstelle des Programm-Managers auf. Die Priorität der Task ist auf 64 voreingestellt.

Programm-Manager

Der Programm-Manager verwaltet alle laufenden CLI-Kommandos.

Zu seinen wesentlichen Aufgaben gehören:

- das Suchen von Programmen über den Suchpfad,
- Eingaben und Ausgaben gegebenenfalls umzulenken,
- die Verwaltung von Hintergrund-Jobs,
- die Verwaltung von mehrfach verwendbaren Code-Segmenten,
- das Abbrechen von Tasks durch <CTRL>+<C> bzw. CANCEL.
- Inline-Kommandos

  Inline-Kommandos werden mit dem Programm-Manager und somit mit dem RMOS-Betriebssystem gebunden. Sie werden in der CLI-Task als Unterprogramme aufgerufen. Die Inline-Kommandos stehen auch ohne Dateiverwaltungssystem (HSFS) zur Verfügung.

Nachladbare Kommandos

Nachladbare Kommandos sind Programme, die über die CLI-Bedienoberfläche aufgerufen werden können. Zum Beispiel sind die Kommandos `dir`, `copy` und `rename` als nachladbare Programme realisiert.

Ablaufumgebung

Die Ablaufumgebung ist ein fester Bestandteil des CLI. Sie

- eröffnet die Ein- und Ausgaben stdin, stdout und stderr unter Berücksichtigung einer eventuellen Umlenkung,
- zerlegt die Kommandozeile in das C-Format (argc/argv),
- initialisiert die CRUN taskspezifisch,

CLISTART.BAT

Beim Starten einer CLI-Session kann eine Batchdatei aufgerufen werden, z.B. um Einstellungen wie Suchpfad, Prompt und aktuelles Verzeichnis zu setzen.

Starten des CLI

Der Start einer CLI-Session kann u.a. durch die INITTASK erfolgen oder durch eine Verteilertask, die den CLI bei einer unerwarteten Eingabe startet. Beide Möglichkeiten sind realisiert. D.h. beim Systemhochlauf wird der CLI an der Hauptkonsole gestartet. An allen Konsolen kann der CLI über die Tastenkombination <CTRL>+<R> gestartet werden.

Beim Start durch die INITTASK ruft der CLI die Datei CLISTART.BAT im Hauptverzeichnis des Bootlaufwerks auf.

Beim Start über die Verteilertask <CTRL>+<R> wird der Name erfragt, der zum Ausführen einer Datei mit der Extension BAT verwendet wird. Wird nur <return> eingegeben, so wird die Standarddatei (CLISTART.BAT) abgearbeitet, die für den Start durch die INITTASK festgelegt wurde.

Gestartete Kommandos und Programme

Vom CLI gestartete Kommandos und Programme werden als Jobs bezeichnet. Sie können im Vordergrund (in Verbindung mit der Konsole) laufen oder im Hintergrund (ohne Verbindung zur startenden Konsole).

Jobs können verschachtelt sein: Ein Job kann mit Batchdateien oder über die Funktionsschnittstelle weitere Jobs starten. Der aufrufende Job wird Vater-Job, der gestartete Job wird Kind-Job genannt. Kind-Jobs können auch weitere Kind-Jobs starten. Alle über den CLI oder die CLI-Funktionsschnittstelle gestarteten Tasks und Kommandos werden vom CLI aus als Jobs verwaltet. Aus Sicht des Nukleus gibt es aber auch Tasks, die über SVC-Aufrufe gestartet werden. Diese sind dem CLI nicht bekannt und somit keine Jobs.

Alle Jobs werden von dem CLI gesteuert, von dem sie gestartet wurden. Der CLI verwaltet u.a. folgende Zuordnungen:

- Jeder Job hat eine Jobnummer, die vom CLI beim Start vergeben wird.
- Jedem Job ist das aktuelle Verzeichnis zugeordnet, das beim Aufruf eingestellt ist. Es wird beispielsweise der Befehl `DIR a:\bin > liste.txt` im aktuellen Verzeichnis A: eingegeben. Die Datei LISTE.TXT wird somit im Verzeichnis A: angelegt oder überschrieben.
- Jedem Job ist eine Standard-Eingabe, Standard-Ausgabe und eine Fehlerausgabe zugeordnet. Die Konsole des CLI ist bei Vordergrund-Jobs für alle Ein- und Ausgaben eingestellt. Den Ein- und Ausgaben von Hintergrund-Jobs wird das NUL-Gerät zugeordnet. Die Zuordnungen können umgelenkt werden (siehe Seite 4-8, 4-9).

Editieren von Kommandos

Eingaben von der Tastatur an die Kommandozeile werden über den BYT-Treiber abgewickelt. Für jede CLI-Session steht ein eigener History-Puffer in der angegebenen Länge zur Verfügung, d.h. zuvor eingegebene Kommandos können über die Pfeiltasten angewählt werden.

Kommandozeilen können mit folgenden Tasten editiert werden:

Buchstaben, Zahlen	überschreiben grundsätzlich bestehende Zeichen
Pfeiltasten	rechts und links erzeugen eine Cursorbewegung
Pfeiltasten	oben und unten kopieren ein zuvor eingegebenes Kommando aus dem Puffer in die Kommandozeile
Einf (INS)	erzeugt ein eingefügtes Leerzeichen
Entf <CTRL>	ersetzt das Zeichen links vom Cursor durch ein Leerzeichen
(Rückschritt)(Backspace)	wie Entf
<CTRL>+<A>	Löscht die Zeile ab Cursorposition bis zum Zeilenende
<CTRL>+<F>	Löscht das Zeichen an der Cursorposition
<CTRL>+<H>	wie Rückschritt

<CTRL>+<X>	Löscht die Zeile vom Anfang bis zur aktuellen Cursorposition
<CTRL>+<Z>	Beendigung der Eingabe beim Kopieren von der Konsole, z.B. Ende des Befehls `copy con datei.txt`.

Eingabe von Namen

Namen dürfen prinzipiell alle Zeichen enthalten, die unter DOS zulässig sind. Das heißt, in den Namen sind alle Zeichen größer ASCII 0x21 zulässig, die keine der folgenden Funktionen haben:

Platzhalterzeichen:

Das Platzhalterzeichen (Joker-Zeichen, Wildcard) "*" steht im Dateinamen für unbestimmte Ergänzungen des Namens bis zur maximal zulässigen Länge.

Das Platzhalterzeichen "?" bezeichnet in Dateinamen genau ein unbestimmtes Zeichen.

Laufwerksangaben:
Laufwerksangaben werden durch das Zeichen ":" abgeschlossen.

Verzeichnisnamen

Verzeichnisnamen müssen von nachfolgenden Datei- oder Verzeichnisnamen durch die Zeichen "/" oder "\" abgetrennt sein. Beide Zeichen sind gleichbedeutend.

Für jeden Job gibt es ein aktuelles Verzeichnis. Pfade und Wege können absolut, d.h. beginnend von der Laufwerksbezeichnung ausgehend angegeben werden oder relativ, d.h. die Struktur wird ausgehend vom aktuellen Verzeichnis angegeben. In diesem Fall repräsentieren die zwei Punkte ".." das übergeordnete Verzeichnis.

Gültige Dateibezeichnungen sind zum Beispiel:

```
C:\CLISTART.BAT
C:CLI\BIN\DIR.386
DIR.386
..\DIR.386
BIN\DIR.386
```

Extensions

Im allgemeinen verwendet der CLI DOS-kompatible Dateinamen. Nachladbare Programme und CLI-Kommandos können ohne Dateierweiterung angegeben werden. Der CLI sucht die Erweiterung in einer Verzeichnisebene in folgender Reihenfolge:

```
386, EXE, BAT
```

Hinweis

DOS-Programme, die als .EXE-Datei vorliegen, können vom CLI nicht ausgeführt werden.

Der CLI wandelt die Dateinamen, Verzeichnisse und Laufwerke in Großbuchstaben. Die Argumente der Kommandozeilen werden jedoch unverändert übergeben. Somit kann ein unter dem CLI laufendes Programm Parameter in Kleinschreibung erhalten.

Umlenkung von Eingaben und Ausgaben

Eingaben und Ausgaben eines Kommandos oder eine Task können auf eine Datei oder eine Schnittstelle umgelenkt werden.

Dies ist möglich bei:

- allen CLI-Kommandos und
- allen Programmen, die durch die Verwendung von Standard-C-Funktionen ihre Eingaben von `stdin` lesen und ihre Ausgaben nach `stdout` oder `stderr` schreiben.

stdout

Eine Umlenkung wird durch eine der folgenden Angaben in der Kommandozeile spezifiziert:

```
> <Datei>
oder
1> <Datei>
```

lenkt alle Ausgaben nach `stdout` auf die angegebene Datei um. Die Datei wird neu angelegt bzw. überschrieben.

```
>> <Datei>
oder
1>> <Datei>
```

lenkt alle Ausgaben nach `stdout` auf die angegebene Datei um. Falls die Datei bereits existiert, werden die Ausgaben an das Ende der Datei geschrieben, andernfalls wird sie neu angelegt.

stderr

Die Angabe

```
2> <Datei>
```

lenkt alle Ausgaben nach `stderr` auf die angegebene Datei um. Die Datei wird neu angelegt bzw. überschrieben.

```
2>> <Datei>
```

lenkt alle Ausgaben nach `stderr` auf die angegebene Datei um. Falls die Datei bereits existiert, werden die Ausgaben an das Ende der Datei geschrieben, andernfalls wird sie neu angelegt.

stdin

Die Angabe

```
< <Datei>
```

veranlaßt, daß Eingaben von `stdin` aus der genannten Datei gelesen werden.

Beispiele:

```
DIR >NUL
DIR > DIR.LIS
```

Anstelle eines Dateinamens kann auch ein Gerätenamen angegeben werden (COM1, LPT1 etc). Ein möglicher Gerätename muß im RMOS-Betriebsmittelverzeichnis Catalog als BYT-Treiber-kompatible Unit eingetragen sein.

Sollen die Ausgaben vollständig unterdrückt werden, bietet sich eine Umlenkung in die Datei NUL an.

Zugreifen auf Laufwerke

Unter dem RMOS-Dateiverwaltungssystem müssen Laufwerke (Volumes) vor der Benutzung angemeldet werden (`MOUNT`). Dieses Anmelden kann auf verschiedene Weise erreicht werden. Das Verhalten ist so, daß das Laufwerk beim **Öffnen** einer Datei automatisch angemeldet wird. Das Laufwerk bleibt dann angemeldet, auch wenn das Öffnen der Datei mit Fehler beendet wurde.

Wird auf ein Laufwerk zugegriffen ohne explizit eine Datei zu öffnen - z. B. wie bei `dir A:` - so muß das Laufwerk mit `MOUNT` angemeldet werden, bevor auf das Laufwerk zugegriffen werden kann.

Das automatische Anmelden eines Laufwerks kann auch durch ein entsprechend gesetzten PATH erreicht werden, wenn ein Kommando überhaupt nicht oder über ein späteres Pfadelement erreicht wird.

Drucken unter M7 RMOS32

Um unter M7 RMOS32 (ohne MS-DOS) eine Datei zu drucken, geben Sie folgendes Kommando im CLI ein:

Drucker an LPT–Schnittstelle:

copy <dateiname> BYT_LPT1

Drucker an COM–Schnittstelle:

copy <dateiname> BYT_COM1

Beschreibungen der CLI-Kommandos

Ausführliche Beschreibungen der CLI–Kommandos finden Sie im Anhang B.

4.3 Programme auf M7-Zielsystem übertragen und löschen

Anwendung

STEP 7 bietet Ihnen die Möglichkeit, über die M7-Verwaltung

- M7-Anwendungsprogramme mit allen zugehörigen Projektdaten entweder einzeln oder zusammen mit dem Betriebssystem auf das M7-Zielsystem zu übertragen
- beliebige Software-Komponenten (M7-Programme) vom M7-Zielsystem zu löschen

M7-Programme auf das M7-Zielsystem übertragen und verwalten können Sie auch mit einigen Entwicklungswerkzeugen, die in der Optionssoftware für M7 enthalten sind, z.B. CFC oder der Organon Debugger (siehe Dokumentation der entsprechenden Optionssoftware). Die Programme werden auf diese Art jedoch nicht permanent auf den Massenspeicher übertragen, sondern nur temporär in den Hauptspeicher des M7-Zielsystems geladen.

Voraussetzung

Um Anwendungsprogramme über MPI auf das M7-Zielsystem übertragen zu können, muß bereits ein Betriebssystem vorhanden sein, mit dem der M7-300/400 hochfahren und eine MPI-Verbindung zum PC/PG aufbauen kann.

Ansonsten können Sie Ihr M7-Anwendungsprogramm auch zusammen mit dem Betriebssystem installieren.

Prinzipielles Vorgehen

Um ein C-, C++-Programm auf das M7-Zielsystem zu übertragen, gehen Sie wie folgt vor:

1. Selektieren Sie einen M7-Programmbehälter, der einer M7-Baugruppe (CPU oder FM) zugeordnet ist.
2. Rufen Sie im Kontext Ihres Projektes den Menübefehl **Zielsystem ▸ M7-Zielsystem verwalten** auf.
3. Öffnen Sie das Register ”Programme”
4. Treffen Sie folgende Auswahlen (siehe Bild 4-3):
   – Programme auf dem Erstellsystem
   – Übertragungs- und Zielmedium
   – Lokales Laufwerk und Partner Laufwerk, wenn Sie das Medium ”MPI/RFS” benutzen (lokales Laufwerk nur unter Windows 95).
5. Betätigen Sie die Schaltfläche ”Installieren”.

Alle weiteren Tätigkeiten hängen von dem ausgewählten Medium ab.

Programme auswählen

Im Listenfeld ”Erstellsystem” werden Ihnen alle C-, C++ Programme aufgelistet, die innerhalb Ihres Projekts dem M7-Zielsystem zugeordnet sind. Sie können eines oder mehrere davon zur Übertragung auswählen. Im Feld ”Zielsystem” werden die bereits auf dem Zielsystem vorhandenen Programme angezeigt.

Übertragungsmedien

Sie haben zwei Wege für die Übertragung der M7-Anwendungsprogramme von STEP 7 aus:

- Online über MPI/RFS
- Offline über Floppy Disk oder Memory Card

Lokales Laufwerk und Partnerlaufwerk auswählen

Wenn Sie das Übertragungsmedium ”MPI/RFS” für die Installation benutzen, so können Sie - wie bei der Betriebssysteminstallation- ein lokales und ein Partnerlaufwerk auswählen (siehe Seite 3-4).

Bild 4-3 Register ”Programme”

Hinweis

Wenn das Betriebssystem auf der Festplatte installiert ist, empfehlen wir aus Gründen der Datensicherheit bei Netzausfall (siehe Kapitel 3.2), die Anwendungsprogramme auf ein anderes Laufwerk als das Betriebssystem zu installieren.

M7-Programme über MPI/RFS übertragen

Bei einer Online-Übertragung werden die notwendigen Programmteile direkt über MPI auf den Massenspeicher des Zielsystems übertragen und die entsprechenden Start-Batch-Dateien werden in die Datei \ETC\INITTAB des Zielsystems eingetragen, so daß die Programme beim nächsten Systemhochlauf automatisch gestartet werden. Zusätzlich wird für jedes M7-Programm eine spezielle Beschreibungsdatei übertragen, die alle notwendigen Informationen zum Anzeigen u. Löschen des Programms enthält. Der Name dieser Beschreibungsdatei wird automatisch aus dem Namen des Programms erzeugt, wobei sichergestellt wird, daß dieser Dateiname immer eindeutig auf dem Zielsystem ist.

Um M7-Programme über MPI/RFS auf das M7-Zielsystem zu übertragen, gehen Sie wie folgt vor:

1. Starten Sie das M7-Zielsystem. Für den Systemhochlauf können Sie auch eine bootfähige Diskette oder Memory Card einsetzen.
2. Starten Sie die M7-Verwaltung mit dem Menübefehl:
   Zielsystem ▸ M7-Zielsystem verwalten
3. Öffnen Sie das Register ”Programme”
4. Treffen Sie folgende Auswahlen (siehe Bild 4-3):
   - Medium: ”MPI/RFS”
   - Lokales Laufwerk (nur Windows 95): das erste freie Laufwerk, z.B. F:
   - Partner Laufwerk: z.B. C: oder D: für Festplatte
   - die gewünschten Anwendungsprogramme aus der ”Erstellsystem”-Liste
5. Wählen Sie die Schaltfläche ”Installieren”.

   Ergebnis: Es wird eine MPI-Verbindung zum M7-Zielsystem aufgebaut und die ausgewählten Software-Komponenten werden auf das Zielsystemlaufwerk übertragen. Die übertragenen Programme werden im ”Zielsystem”-Feld angezeigt.

   Die Programme werden beim nächsten Systemstart automatisch gestartet.

Bei der Ausführung dieser Schritte werden Sie im Dialogfeld über das aktuelle Geschehen informiert.

Vorsicht

Falls auf dem Zielsystem gleichnamige Dateien vorhanden sind, werden diese beim Übertragungsvorgang überschrieben! Es gibt kein automatisches Umbenennen und kein automatisches Backup!

M7-Programme über Datenträger übertragen

Bei einer offline Übertragung werden zunächst alle Dateien auf eine Diskette bzw. Memory Card kopiert. Zusätzlich wird auf dem Datenträger eine Installationsdatei **m7swins.bat** angelegt, mit dessen Hilfe die zuletzt selektierten Programme von der Diskette bzw. Memory Card auf die Massenspeicher des M7-Zielsystems übertragen werden können. Die Datei m7swins.bat muß unter dem CLI von M7 RMOS32 ausgeführt werden.

Hinweis

Bei offline Übertragung über Memory Card auf ein M7–Zielsystem mit M7 RMOS32/DOS müssen Sie in der Datei **m7swins.bat** den Eintrag

M7INSTDRIVE=M0:

ändern in M7INSTDRIVE=B:

Um M7-Programme offline mittels eines Datenträgers zu übertragen, gehen Sie wie folgt vor:

1. Starten Sie die M7-Verwaltung mit dem Menübefehl:
   Zielsystem ▸ M7-Zielsystem verwalten
2. Legen Sie den Datenträger in das Laufwerk des PC/PG ein.
3. Öffnen Sie das Register ”Programme”
4. Treffen Sie folgende Auswahlen (siehe Bild 4-3):
   – ”Floppy Disk” oder ”Memory Card” als Medium und
   – die gewünschten Anwendungsprogramme aus der Auswahlliste ”Erstellsystem”
5. Betätigen Sie die Schaltfläche ”Installieren”

   Ergebnis: Die ausgewählten Software-Komponenten werden auf den Datenträger übertragen.
6. Legen Sie den Datenträger in das M7-Zielsystem ein.
7. Starten Sie den CLI lokal am M7-300/400 oder über Remote Terminal.
8. Rufen Sie das auf dem Datenträger befindliche Batch-File **m7swins.bat** auf, um die Software-Komponenten auf die Festplatte zu kopieren. Das Batch-File m7swins.bat kopiert immer in das gerade aktive Laufwerk. D.h. für eine Übertragung von Diskette auf die Festplatte geben Sie z.B. folgende Befehlsfolge ein:

   ```
   cd c:\
   A:\m7swins.bat
   ```

Vorsicht

Falls auf dem Zielsystem gleichnamige Dateien vorhanden sind, werden diese beim Übertragungsvorgang überschrieben!

9. Die Batch-Dateien zum Start der Programme werden nicht automatisch in die \ETC\INITTAB des M7-Zielsystems eingetragen. Stattdessen wird eine temporäre Datei \ETC\INITTAB.ins auf dem Datenträger angelegt, die alle notwendigen Einträge enthält. Um die Programme automatisch beim nächsten Systemstart des M7-300/400 zu starten, müssen Sie diese Einträge mit Hilfe eines Editors in die Datei \ETC\INITTAB des M7-Zielsystems übernehmen.

Regeln für die Offline-Übertragung

Wenn Sie M7-Programme offline über Datenträger auf das Zielsystem übertragen, wird für jedes M7-Programm eine spezielle Beschreibungsdatei auf dem Datenträger hinterlegt, die alle notwendigen Informationen zum Anzeigen u. Löschen des Programms enthält. Der Name dieser Beschreibungsdatei wird automatisch aus dem Namen des Programms erzeugt. Um sicher zu stellen, daß dieser Dateiname immer eindeutig auf dem Zielsystem ist, muß eine der folgenden Bedingungen erfüllt sein:

- Die Namen der für eine M7-CPU bzw. M7-FM definierten Programme unterscheiden sich in den ersten 5 Zeichen.
- Es werden stets alle zu einer CPU bzw. FM gehörenden Programme auf den Datenträger kopiert und von dort auf das M7-Zielsystem übertragen.

Hinweis

Wenn dies nicht beachtet wird, besteht die Gefahr, daß bei einem späteren Zugriff über MPI eine Software-Komponente nicht mehr in der ”Zielsystem”-Auswahlliste angezeigt wird und somit auch nicht mehr gelöscht werden kann.

M7-Programme löschen

Um M7-Programme online von dem M7-Zielsystem zu löschen, gehen Sie wie folgt vor:

1. Führen Sie die Schritte 1. bis 4. wie bei der MPI/RFS-Übertragung von Anwendungsprogrammen aus.
2. Wählen Sie die Software-Komponenten, die gelöscht werden sollen, aus der ”Zielsystem”-Liste aus.
3. Betätigen Sie die Schaltfläche ”Löschen”.

   Ergebnis: Die ausgewählten Software-Komponenten werden von dem Zielsystemlaufwerk gelöscht.

4.4 Starten der Anwendungsprogramme auf dem M7-Zielsystem

M7-Programme starten

Sie haben folgende Möglichkeiten, Anwendungsprogramme auf dem M7-Zielsystem zu starten:

1. Im laufenden Betrieb per Bedienung über die lokale Konsole oder über Remote Terminal Interface (RTI).
2. Beim Systemhochlauf, durch einen Eintrag in der Datei \ETC\INITTAB. Diese Datei wird gleich nach dem Hochlauf des Betriebssystems gelesen. Sie enthält die Aufrufe aller Programme, die automatisch beim Systemstart ausgeführt werden sollen.

   Wenn Sie die Anwendungsprogramme zusammen mit dem Betriebssystem oder über ”MPI/RFS” installieren, werden sie automatisch in die Datei \ETC\INITTAB eingetragen.

   Wenn Sie die Programme ohne Betriebssystem über Datenträger (Diskette oder Memory Card) übertragen, müssen Sie die Einträge selbst vornehmen, wenn die Programme beim Systemstart automatisch ausgeführt werden sollen. Die passenden Einträge finden Sie in der Datei \ETC\INITTAB.INS.

Starten und abbrechen der Programme über CLI

Innerhalb einer CLI-Session können Sie ein Anwederprogramm starten indem Sie das Programm in der Kommandozeile aufrufen, z.B.:

- c:\testprog

Das gestartete Programm kann durch die Eingabe von ***<CTRL> C*** beendet werden.

Vorsicht

Programme, die mit ***<CTRL> C*** beendet werden, geben belegte Ressourcen nicht wieder frei. Dies kann zum Absturz der Systemserver führen.

Bitte sorgen Sie im Programm dafür, daß der Programmabbruch verhindert wird (siehe Aufruf **inhibitabort**), und daß möglichst alle belegten Ressourcen wieder freigegeben werden.

Gegebenenfalls starten Sie das System erneut, bevor Sie das abgebrochene Programm erneut starten.

4.5 Abfrage- und Steuerfunktionen für M7-300/400

Auskunftsfunktionen

Mit dem Menübefehl **Zielsystem ▸ Baugruppenzustand** können Sie folgende Informationen über M7-Zentralbaugruppen am PC/PG abfragen:

- Zeitsystem und CPU-Zeit
- M7-CPU-Daten
- Auslastung des Anwenderspeichers
- Zykluszeiten der CPU
- Status der Kommunikationsverbindungen
- Inhalt des Diagnosepuffers

Was ist anders

Anders als bei S7-Baugruppen, können Sie für M7-CPU folgende Informationen **nicht** abfragen:

- Bausteindaten und
- Stack-Inhalte

Die entsprechenden Register und Felder sind an der Benutzungsoberfläche zwar zugänglich, aber ohne Inhalt.

CPU-Meldungen

Mit der Funktion **"CPU Melden"** können asynchrone Meldungen von Fehlerereignissen und benutzerdefinierte Meldungen ausgegeben werden.

Einstellungen

Folgende Einstellungen können Sie an M7-CPU - wie an S7-CPU - vornehmen :

- Betriebszustand ändern, CPU urlöschen
- Uhrzeit einstellen

Variable abfragen und steuern

Mit dem Menübefehl **Zielsystem Variable beobachten/steuern** stehen folgende Funktionen zur Bearbeitung der Variablentabelle zur Verfügung:

- Lesen der Inhalte von Datenbausteinen, Eingängen, Ausgängen und Merkern.
- Schreiben der Inhalte von Datenbausteinen, Eingängen, Ausgängen und Merkern.

Hinweis

Das Forcen von Variablenwerten wird auf SIMATIC M7 nicht unterstützt.

Im Betriebszustand STOP ist das Steuern von Ausgängen nicht möglich, weil das OD-Signal nicht ausgeschaltet werden kann. Gehen Sie dazu in den Betriebszustand RUN.

Wo Sie Informationen finden

Ausführliche Informationen zu den o.g. Funktionen finden Sie:

- im STEP 7 Benutzerhandbuch und
- in der Online-Hilfe zu STEP 7

Arbeiten mit Massenspeichern

5

In diesem Kapitel

In diesem Kapitel erfahren Sie:

- wie Sie Memory Cards und OSDs unter M7 RMOS32 und MS-DOS formatieren
- wie Sie Festplatten und Disketten unter M7 RMOS32 partitionieren und initialisieren

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
5.1	Formatieren der Memory Cards und OSD	5-2
5.2	Formatieren der Fesplatte und Diskette	5-3
5.3	Festplattenpartitionierungsprogramm HDPART	5-4
5.4	Formatierung von Memory Cards und OSDs unter MS-DOS	5-9
5.5	Arbeiten mit Memory Cards	5-11
5.6	Programm REMAP_A	5-12

5.1 Formatieren der Memory Cards und OSD

Wie werden Memory Cards und OSD formatiert?

Für die Formatierung von Memory Cards und OSD stehen folgende Befehle zur Verfügung:

Unter MS-DOS auf M7-300/400:
benutzen Sie den Befehl **ftlform.exe** (siehe Seite 5-9).

Unter MS-DOS auf PG:
benutzen Sie den Befehl

- **step7\s7bin\s7oformx.exe** unter Windows 95
- **step7\s7bin\s7ofornx.exe** unter Windows NT

(siehe Seite 5-9).

Unter M7 RMOS32:
benutzen Sie den Befehl **bin\ftlform.386** (siehe Seite B-22).

Diesen Befehl können Sie nur am M7-Zielsystem lokal oder über Remote-Terminal aufrufen.

Die Befehle haben die folgende Syntax:
ftlform <drive_to_format:> [/option]

s7oformx <drive_to_format:> [/option] bzw.
s7ofornx <drive_to_format:> [/option]

wobei die Optionen unter MS-DOS und M7 RMOS32 unterschiedliche Bedeutungen haben.

Memory Cards können Sie sowohl auf dem PC/PG als auch auf dem M7-Zielsystem formatieren.

Onboard Silicon Disks (OSD) können Sie lokal an dem M7-Zielsystem oder über Remote-Terminal formatieren.

Formatieren unter MS-DOS

Unter MS-DOS wird nach der Option **/s** das Laufwerk angegeben, von welchem die System-Dateien auf das Bootmedium kopiert werden sollen.

Beispiele:
Formatierung der Memory Card, Laufwerk B: als Bootmedium (die MS-DOS-Installationsdiskette befindet sich in Laufwerk a:):
`s7oformx b: /s a:` im DOS-Fenster des PC/PG

Einfache Formatierung der OSD auf dem M7-Zielsystem
`ftlform <Laufwerk:>`

`<Laufwerk:>` ist das zu formatierende Laufwerk. In der Regel ist der OSD das erste freie Laufwerk nach den Festplattenpartitionen zugeordnet (siehe Tabelle 3-2 auf Seite 3-5).

Formatieren unter M7 RMOS32

Unter M7 RMOS32 wird nach der Option **/s** die System-Datei angegeben, welche auf das Bootmedium kopiert werden soll. Ist keine Datei angegeben, wird per Voreinstellung die Datei RM3_PC1.SYS kopiert. Der Befehl muß aus dem Root-Verzeichnis (C:\) aufgerufen werden, auf dem sich diese Datei befindet.

Beispiel - Formatierung der Memory Card als Bootmedium:

```
bin\ftlform M0: /s
```

5.2 Formatieren der Festplatte und Diskette

Wie werden Festplatten und Disketten formatiert?

Für die Formatierung von Festplatten und Disketten steht der Befehl **format** zur Verfügung. Dieser Befehl kann sowohl unter M7 RMOS32 als auch unter MS-DOS, bzw. im DOS-Fenster von Windows 95 auf dem PC/PG aufgerufen werden und hat in beiden Fällen die gleiche Syntax. Im Anhang B finden Sie eine Beschreibung des Befehls unter M7 RMOS32.

Nachfolgend wird nur die Formatierung unter M7 RMOS32 beschrieben. Die Beschreibung für MS-DOS entnehmen Sie bitte dem MS-DOS-Handbuch.

Hinweis

Ab Version V 2.0 der Systemsoftware für M7-300/400 können unter M7 RMOS32 Festplatten mit einer Kapazität > 516 x 10^6 byte verwaltet werden.

Erstmaliges Formatieren der Festplatte

Vor jeder Installation von M7 RMOS32 (ohne MS-DOS) auf die Festplatte des M7-Zielsystems ist es erforderlich, die Bootpartition der Festplatte neu zu formatieren. Um eine Festplatte erstmalig unter M7 RMOS32 als Bootmedium zu formatieren, gehen Sie wie folgt vor.

1. Starten Sie den CLI lokal am M7-300/400 oder über Remote Terminal Interface.
2. Geben Sie folgende Kommandos ein:

   `HDPART` (siehe Abschnitt 5.3) und starten Sie das M7-Zielsystem erneut

   `FORMAT C:` (siehe Abschnitt B.15)

   `FORMAT D:`

   ... (wenn mehrere Partitionen vorhanden sind)

   `RDISK` (siehe Abschnitt B.25)

Kommando HDPART

Dieses Kommando bietet Ihnen folgende Menüs an:

1. `Add partition`
2. `Change partition` (tragen Sie in der Spalte ”Status” das Zeichen **B** ein, um die Partition als Bootpartition zu kennzeichnen)
3. `Accept partition table`

Zusätzliche Informationen finden Sie im Abschnitt 5.3.

Nachträgliches Formatieren der Festplatte

Ist bereits ein logische Partition auf der Festplatte angelegt, so können die Partitionierung und Initialisierung mit `RDISK` entfallen. Gehen Sie wie folgt vor.

1. Starten Sie den CLI lokal am M7-300/400 oder über Remote Terminal Interface.
2. Geben Sie folgende Kommandos ein:

   `DISMOUNT` (Dieser Schritt ist nur erforderlich, wenn die Partition, die Formatiert werden soll, gemountet ist; siehe Abschnitt B.11)

   `FORMAT C:` (siehe Abschnitt B.15)

   `RDISK RM_PC1.SYS` (siehe Abschnitt B.25)

Formatieren von Disketten

Disketten werden mit dem CLI-Kommando **FORMAT** formatiert

(siehe Abschnitt B.15).

5.3 Festplattenpartitionierungsprogramm HDPART

Mit dem Festplattenpartitionierungsprogramm HDPART können Sie Ihre Festplatte partitionieren.

Starten

Unter dem CLI muß nur der Dateiname und der Pfad zum Programm eingegeben werden, um das Programm zu starten, z.B.:

```
C:BIN\HDPART
```

Bedienerführung

Das Programm ist menügeführt. Ein Eingabemenü besteht aus Eingabefeldern. Zum jeweils nächsten Eingabefeld kann mit <TAB> gewechselt werden. Ein Eingabefeld kann mit der Rücktaste editiert und mit <Ctrl>+<X> gelöscht werden. Mit <Esc> werden die Eingaben im aktuellen Menü verworfen und zum vorhergehenden Menü zurückgekehrt.

Ablauf der Partitionierung

Nach dem Aufruf meldet sich HDPART z.B. wie folgt:

```
(c) Siemens AG, RMOS3 Partdisk, Harddisk Partition Program Vx.y
Selected: Device 04H, Unit 00H
Work on this Unit (y/n) ?
```

Wenn ein Zugriff auf die Unit des Festplattentreibers nicht möglich ist, wird eine neue Unit-ID angefordert. Andernfalls wird die Bestätigung für diese Unit verlangt oder es kann eine andere Unit gewählt werden.

Geben Sie einen der angegebenen Auswahlwerte gefolgt von <Return> ein. Ihre Eingabe führt Sie zu folgenden Untermenüs:

Hauptmenü:

Die Partitionierungsdaten werden in Form einer Tabelle angezeigt.

Eine Tabellenzeile bezieht sich jeweils auf eine Partition. Sie enthält Art, Status, erste LBA (logische Blockadresse), letzte LBA, ersten Zylinder, letzten Zylinder und die belegte Plattenkapazität (in Prozent) der Partition. Haben Sie noch keine Partition angelegt, wird nur die Kopfzeile der Tabelle ausgegeben.

```
Part-No|Status|Type |1st LBA|Last LBA|1st Cyl|Last Cyl|% of all
-------+------+-----+-------+--------+-------+--------+--------
     1 |    B |DOS-4|     64|   56944|      0|     203|  20%
     2 |      |DOS-4|  56960|   85392|    203|     304|  10%
     3 |      |DOS12|  85408|  100000|    305|     357|   5%
     4 |      |DOS-3| 100016|  150000|    357|     535|  17%
1. Accept Partition Table
2. Change Partition Table
3. Add one Partition
4. Abort without Changes
Enter Selection:
```

Die Bedeutung der einzelnen Werte ist im Kapitel ”Aufbau der Partitionstabelle” beschrieben. Geben Sie einen der angegebenen Auswahlwerte gefolgt von <Return> ein. Ihre Eingabe führt Sie zu folgenden Untermenüs:

Menü: Accept

```
1           Accept Partition Table
```

Die am Bildschirm angezeigten Partitionswerte, sind in Ordnung und Sie wollen diese Partitionierung akzeptieren. Es folgt eine Sicherheitsabfrage, ob sie die Daten auf die Platte schreiben wollen.

```
Write changes to disk (y/n) ?
```

Wenn Sie n eingeben, wird die Menüfunktion beendet, ohne daß die Daten auf die Platte geschrieben werden. Bestätigen Sie die Abfrage mit y, werden die für die automatische Festplattenerkennung (durch x_hd_init) notwendigen Daten auf die Platte geschrieben und die Funktion beendet.

```
Program terminated normally
```

Hinweis

Erst dieser Befehl schreibt die in den Untermenüs ”2 Change Partition Table” und ”3 Add one Partition” getroffenen Einstellungen auf die Festplatte zurück.

Menü: Change

```
2           Change Partition Table
```

In diesem Untermenü können Sie die Daten einer Partition ändern.

Hinweis

Die Partition, die geändert werden soll, darf nicht mit `MOUNT` angemeldet sein, vor dem Aufruf von `HDPART` müssen Sie sie mit dem CLI-Kommando `DISMOUNT` abmelden.

Jeweils eine Zeile der Partitionstabelle wird ausgegeben.

```
Part-No|Status|Type |1st LBA|Last LBA|1st Cyl|Last Cyl|% of all
-------+------+-----+-------+--------+-------+--------+-------
     1 |    B |DOS-4|     64|   56944|      0|     203|  20%
```

In dieser Zeile können Sie Änderungen durch einfaches Überschreiben der ausgegebenen Werte vornehmen. Wird von einem Feld zum anderen (mittels <TAB>) gewechselt, werden die abhängigen Parameter entsprechend modifiziert, jedoch keine Verifikation durchgeführt. Mit <Return> wird die nächste Partition zum Editieren ausgegeben.

Ist keine weitere Partition vorhanden und sind keine freien Bereiche mehr auf der Platte, werden alle Partitionsdaten aufgelistet, und in das vorhergehende Menü zurückgekehrt.

Sind auf der Festplatte noch keine Partitionen angelegt, wird die ganze Platte als eine Partition angeboten. In der Spalte `Part-No` steht vor der Nummer der Partition zusätzlich der Buchstabe N, der eine neue Partition kennzeichnet. Die Größe dieser Partition können Sie jetzt verkleinern, indem Sie die Werte in den Spalten `1st LBA`, `Last LBA`, `1st Cyl` oder `Last Cyl` modifizieren. Haben Sie die gewünschte Größe eingetragen und die Eingabe mit <Return> abgeschlossen, wird im nächsten Eingabefeld der Rest der Festplatte als Partition angeboten.

Ist die Festplatte bereits partitioniert, müssen Bereiche, die neu vergeben werden sollen, zuerst gelöscht werden. Eine Partition wird gelöscht, indem Sie bei Spalte Type die Zahl 0 gefolgt von <Return> eingeben.

Sind alle Bereiche vergeben oder wird <Esc> eingegeben, wird die gesamte Partitionstabelle ausgegeben. Dabei wird überprüft ob sich einzelne Partitionen überlappen. Ist dies der Fall, steht vor den entsprechenden Partitionsnummern der Buchstabe o. Das bedeutet, daß Sie noch einmal dieses Menü auswählen und die Partitionsdaten entsprechend anpassen müssen.

Menü: Add

`3 Add one Partition`

Über dieses Untermenü können Sie eine neue Partition anlegen. Die erste gefundene Lücke auf der Festplatte wird Ihnen als neuer Partitionseintrag angeboten. Ist die maximal zulässige Anzahl von Partitionen erreicht (maximal 4) oder sind keine freien Bereiche mehr auf der Festplatte vorhanden, wird das Menü nicht angewählt.

Menü: Abort

`4 Abort without Changes`

Durch diese Eingabe wird die Systemfunktion beendet, ohne daß die in ”2 Change Partition Table” oder ”3 Add one Partition” vorgenommenen Änderungen auf der Platte gesichert werden.

Aufbau der Partitionstabelle

Die einzelnen Felder der Partitionstabelle, die vom Programm HDPART ausgegeben wird, haben folgende Bedeutung:

Tabelle 5-1 Aufbau der Partitionstabelle

Feld	Bedeutung
Part-No	Laufende Nummer der Partitionen, beginnend mit 1. Dieses Feld ist nicht editierbar.
Status	Dieses Feld enthält den Status der Partition, wobei B für Bootpartition steht. Durch Eingabe von einem Leerzeichen wird die aktuelle Partition als nicht bootfähig gekennzeichnet.
Type	In diesem Feld wird festgehalten, um welche Art von Partition es sich handelt. Zulässig sind folgende Werte: DOS-3 DOS Partition < 32 Mbyte mit 16-Bit-FAT-Einträgen DOS-4 DOS Partition > 32 Mbyte mit 16-Bit-FAT-Einträgen DOS12 DOS Partition < 32 Mbyte mit 12-Bit-FAT-Einträgen EXTND Extended-Partition (nur bei HD0 möglich; kann in einer Partitionstabelle nur einmal vorkommen) 0 Löschen der aktuellen Partition; Eingabe nur an erster Eingabeposition gefolgt von <Return>
1st LBA	Erste LBA der Partition
Last LBA	Letzte LBA der Partition
1st Cyl	Erster Zylinder der Partition
Last Cyl	Letzter Zylinder der Partition
% of all	Größe der Partition im Verhältnis zur gesamten Plattengröße

5.4 Formatierung von Memory Cards und OSDs unter MS-DOS

Syntax

Zur Formatierung von Flashlaufwerken dienen die Programme:

- FTLFORM.EXE auf M7-Zielsystemen
- S7OFORMX.EXE auf PG/PC mit Windows 95 oder S7OFORNX.EXE auf PG/PC mit Windows NT

(das DOS-Kommando FORMAT wird von den Laufwerken nicht unterstützt).

Da die beiden Befehle die gleiche Syntax haben, wird nachfolgend nur noch FTLFORM beschrieben.

Der Aufruf erfolgt auf Kommandozeilenebene:

```
c:>FTLFORM X: /Optionen
```

wobei X: das zu formatierende Laufwerk bezeichnet. Beim Aufruf ohne Parameter oder mit den Parametern /h oder /? wird ein Hilfemenue angezeigt:

```
USAGE: FTLFORM <drive:> [/option[sclmnb]]
```

Optionen

Die Optionen können mit "-" oder mit "/" eingeleitet werden, bei der Angabe wird nicht zwischen Groß- und Kleinschreibung unterschieden. Alle Optionen sind durch Leerzeichen voneinander zu trennen. Folgende Optionen werden von dem Programm unterstützt:

- `/b` Batchmode. Die Eingabeaufforderungen des Programmes an den Benutzer werden unterdückt.
- `/c<Dateiname>` Formatierung mit Konfigurationsdatei. Wird kein Dateiname angegeben, versucht das Programm, die Datei FLTFORM.CFG als Konfigurationsdatei zu öffnen; ist das nicht möglich, wird mit einer Fehlermeldung abgebrochen. Die für die Konfigurationsdatei zu verwendende Syntax wird im nächsten Abschnitt erläutert.
- `/<lDateiname>` Formatierung mit Ausgabedatei. Alle Ausgaben des Programmes werden in eine Datei umgeleitet. Wird kein Dateiname angegeben, erfolgt die Ausgabe in die Datei FLTFORM.LOG. Alle Eingabeaufforderungen des Programmes an den Benutzer werden unterdrückt.
- `/m` Anlegen einer Partitionstabelle (Master Boot Record) . Bei der Formatierung einer On-Board Silicon Disk wird grundsätzlich eine Partitionstabelle angelegt.
- `/n` – No Messages. Alle Bildschirmausgaben und Eingabeaufforderungen des Programmes an den Benutzer werden unterdrückt.
- `/s<Laufwerk>` Kopieren der DOS-Systemdateien. Wird kein Laufwerk angegeben, werden die Systemdateien zuerst auf Laufwerk C:, dann auf Laufwerk A: gesucht.
- `/?` oder `/h` Ausgabe des Hilfemenues.

Konfigurationsdatei

Die Konfigurationsdatei ist eine ASCII-Datei, welche eine benutzerspezifische Formatierung von Flashmedien ermöglicht. Die Untergliederung der Datei erfolgt in Sektionen, wobei jede Sektion mit einem Sektionsnamen eingeleitet wird und mit dem Beginn einer neuen Sektion oder dem Ende der Datei beendet wird.

Kommentarzeilen beginnen mit dem Symbol ”#” und können an beliebiger Stelle der Datei eingefügt werden.

Systemfiles

Überschrift: [SYS_FILES]

Diese Sektion enthält die Namen der Systemdateien, welche auf das Medium kopiert werden sollen. Dabei ist für jeden Dateinamen eine gesonderte Zeile zu verwenden. Es können maximal drei Systemdateien spezifiziert werden.

Bootstraploader

Überschrift: [BOOT_RECORD]

Diese Sektion enthält den Namen einer Binärdatei, welche als Bootstraploader auf das Medium kopiert werden soll (Sektor 0 der Partition). Das Programm prüft, ob die Datei den Minimalanforderungen an einen Bootstraploader genügt (Dateilänge 512 Byte, Kennung 55AAh am Ende der Datei).

Die Ablauffähigkeit des Codes wird dabei nicht geprüft!

Partitionstabelle

Überschrift: [MASTER_BOOT_RECORD]

Diese Sektion enthält den Namen einer Binärdatei, welche als Partitonstabelle auf das Medium kopiert werden soll (Sektor 0 des Mediums).

Das Programm prüft, ob die Datei den Minimalanforderungen an einen Masterbootrecord genügt (Dateilänge 512 Byte, Kennung 55AAh am Ende der Datei). Die Ablauffähigkeit des Codes wird dabei nicht geprüft!

Applikationskennung

Die Applikationskennung ”M7-DOS” wird nach erfolgter Formatierung in den Kennbitspeicher der Memory-Card eingetragen.

5.5 Arbeiten mit Memory Cards

Ziehen/Stecken der Memory Card

Wenn Sie die Memory Card im laufenden Betrieb ziehen und wieder stecken wollen, müssen Sie folgendermaßen vorgehen:

1. Wechseln Sie in das Rootverzeichnis der Memory Card M0:\.
2. Führen Sie den Befehl **dismount** für die Memory Card aus. Damit stellen Sie sicher, daß keine Zugriffe auf die Memory Card stattfinden. Wenn der **dismount** erfolgreich war, können Sie die Memory Card ziehen.

Nach dem Stecken der Memory Card wird das Rootverzeichnis automatisch gemountet.

Hinweis

Ein Urlöschen wird nach dem Ziehen/Stecken der Memory Card vom Betriebssystem **nicht** angefordert.

Permanenter Ladespeicher auf Memory Card

Wenn sich der permanente Ladespeicher (Verzeichnis \RAMDIR) auf der Memory Card befindet, darf die Memory Card im laufenden Betrieb nicht gezogen werden, da ansonsten bei einem Ausfall der Netzspannung Konfigurationsdaten verlorengehen können.

5.6 Programm REMAP_A

Funktion

Floppy A: unter MS-DOS wieder zugänglich machen

Aufruf

Der Aufruf erfolgt **unter MS-DOS** ohne Parameter:

```
A:>REMAP_A
FTL100 Remap Drive A: Utility Px.yy
       Copyright SCM Microsystems GmbH 1995
       Drive A: remapped
B:>
```

Beschreibung

In einem System mit vorhandenem Floppylaufwerk A: prüft die BIOS-Erweiterung für Memory-Card und OSD während der Initialisierung, ob sich eine Diskette im Laufwerk befindet. Ist das nicht der Fall, übernimmt bei gesteckter Memory-Card das Laufwerk B: alle Zugriffe auf das Laufwerk A:, um ein Booten von der Memory-Card zu ermöglichen.

Nach dem Booten aber muß dem System das Floppylaufwerk wieder zugänglich gemacht werden. Dies geschieht mit dem Programm REMAP_A.EXE.

Das Programm wechselt selbstständig das aktuelle Laufwerk nach B:, wenn man sich bei dessen Aufruf auf A: befindet.

Hinweis

Der Aufruf ist per Voreinstellung unter M7 RMOS32 mit MS-DOS immer in die Datei AUTOEXEC.BAT eingetragen.

Low-Level-Debugger

6

In diesem Kapitel

In diesem Kapitel erfahren Sie:

- Wie der Low-Level-Debugger funktioniert
- Wie Sie mit dem Low-Level-Debugger arbeiten

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
6.1	Betriebsarten Task-Modus und Monitor-Modus	6-2
6.2	Bedienung des Debuggers	6-5
6.3	Allgemeine Syntax-Regeln	6-7
6.4	Kurzbeschreibung der Debugger-Kommandos	6-9

6.1 Betriebsarten Task-Modus und Monitor-Modus

Der Debugger läuft standardmäßig im Task-Modus, der Betriebsart für das Testen von Programmen auf Taskebene. In dieser Betriebsart dürfen Haltepunkte nur auf Taskebene gesetzt werden, nicht im Code von RMOS-Treibern.

Das Testen von RMOS-Treibern ist nur im Monitor-Modus möglich.

Das Umschalten in die gewünschte Betriebsart erfolgt mit einem entsprechenden Kommando bzw. mit einer speziellen Breakpoint-Behandlung.

Der Debugger enthält außerdem eine Schnittstelle zum Anschluß des CADUL Debuggers Organon XDB.

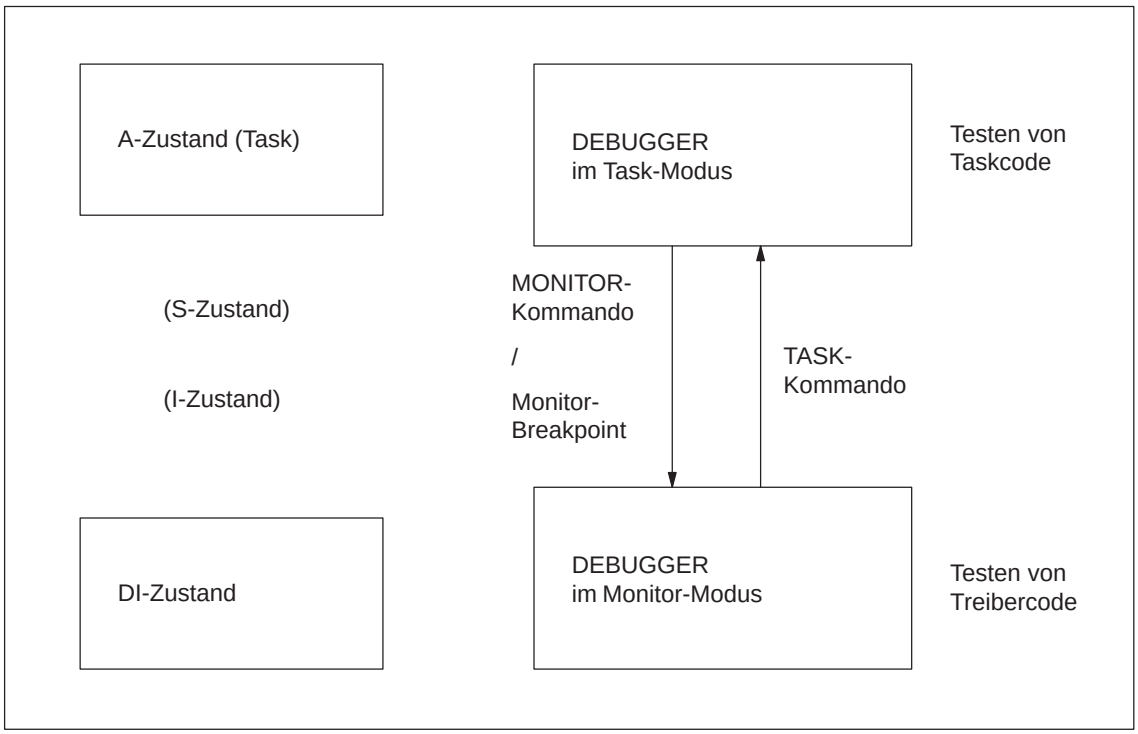

Bild 6-1 Task-Modus/Monitor-Modus

Debugger-Prompt

Der Debugger-Prompt zeigt die aktuelle Betriebsart und den aktuellen Kontext an:

Prompt	Betriebsart	
	Monitor-Modus	**Task-Modus**
Escape-Kontext:		DEBUG_T>
Breakpoint-Kontext	DEBUG_MB>	DEBUG_TB>
Prompt für Host-Debugger-Einsatz	**>**	**>**

Zunächst wird aus Kompatibilitätsgründen nur eine verkürzte Prompt-Meldung **>** ausgegeben.

Mit dem HELP-Kommando kann von der verkürzten Prompt-Meldung > auf die ausführliche Prompt-Meldung, z.B. DEBUG_T> oder DEBUG_TB>, umgeschaltet werden.

Die Umschaltung zurück auf die verkürzte Prompt-Meldung > erfolgt durch Eingabe von speziellen Kommandos, die vom Host-Debugger gesendet werden, z.B. durch Eingabe von v oder V.

Task-Modus

Der Task-Modus ist die Betriebsart zum Testen auf Task-Ebene. Sie ist die standardmäßige Betriebsart des Debuggers. Der Debugger läuft als eigene Task und untersteht der Steuerung durch das Betriebssystem. Über den Scheduler wird ihm CPU-Zeit zugeteilt.

Einzelne Tasks können durch das Setzen von Breakpoints angehalten werden, ohne daß andere Tasks unmittelbar davon beeinflußt werden (Echtzeitverhalten des Debuggers).

Wenn eine Task auf einen Breakpoint läuft, wird sie in den Zustand ”Blocked, debugger breakpointed” überführt und die Debugger-Task wechselt in den Breakpoint-Kontext. Von der unterbrochenen Task können die Register, der Stack und alle benötigten Speicherbereiche angezeigt und wenn nötig verändert werden.

Aus dem Task-Modus kann entweder über einen Monitor-Breakpoint (siehe QUALIFY) oder durch das Kommando MONITOR in den Monitor-Modus geschaltet werden.

Monitor-Modus

Im Monitor-Modus übernimmt der Debugger als Monitorprogramm die Steuerung über die Applikation. Sobald der Debugger im Monitor-Modus läuft, sind alle Interrupts gesperrt und alle weiteren Aktivitäten des Betriebssystems eingestellt, d.h. sämtliche Tasks sind angehalten (kein Echtzeitverhalten).

Nach dem Erreichen eines Monitor-Breakpoints kann keine Taskumschaltung zum Debugger erfolgen. Stattdessen wird der Debugger als Funktion des unterbrochenen Programmes aufgerufen. Der Stack des unterbrochenen Programmes bleibt unbelastet, da auf den Stack des Debuggers umgeschaltet wird. Der Status des unterbrochenen Programmes wird in einer Breakpoint-Struktur (Schattenregister) gespeichert und kann während der Laufzeit des Debuggers angezeigt und modifiziert werden.

Mit dem Verlassen des Debuggers (GO/STEP/EXITK) werden die Registerwerte aus der Breakpoint-Struktur (Schattenregister) übernommen, die Interrupts freigegeben und das unterbrochene Programm fortgesetzt. Ab diesem Zeitpunkt gehört die Steuerung wieder dem Betriebssystem.

Im Monitor-Breakpoint-Kontext kann mit dem Kommando TASK nicht in den Task-Modus geschaltet werden. Erst mit einem folgenden GO-Kommando wird die Programmbearbeitung fortgesetzt und der Monitor-Modus beendet. Der Debugger befindet sich anschließend wieder im Task-Modus.

Anmerkungen:

- Im Monitor-Modus sind taskspezifische Kommandos nicht mehr erlaubt. Da die Interrupts gesperrt sind, wird die Schnittstelle für die Kommunikation im Polling-Verfahren bedient.
- Der Monitor-Modus ist nur an einer seriellen Schnittstelle verfügbar und kann nicht an einer EGA-Unit des BYT-Treibers verwendet werden.
- Der Debugger eignet sich nicht zum Testen von Treibern, die auf denselben seriellen Schnittstellencontroller zugreifen, den auch der Debugger verwendet.
- Der Debugger kann im Task-Modus zu jedem Zeitpunkt von einem Monitor-Breakpoint unterbrochen werden.

  Wenn der Debugger während der Ausgabe des Prompts von einem Monitor-Breakpoint unterbrochen wurde, wird nach einem GO-Kommando im Monitor-Modus die Prompt-Meldung unvollständig ausgegeben.
- Beim Testen von Treibern im Monitor-Modus dürfen keine DATA ACCESS Breakpoints auf die Systemvariablen wie XCIRB, XUCB gesetzt werden, da auf diese Variablen auch vom Debugger zugegriffen wird.

Hinweis

Der Debugger greift erst in den Systemablauf ein, wenn die Aufforderung dazu gegeben wird. Die zu testenden Tasks sollten im RAM liegen, da schneller Programmänderungen durchgeführt und uneingeschränkt Breakpoints gesetzt werden können.

Die INT-3-Anweisung (einzige 1-Byte-Interrupt-Instruktion der CPU) und der Single-Step-Interrupt (INT 1) werden für Debugger-Breakpoints verwendet.

6.2 Bedienung des Debuggers

Bedeutung von Kontext

Für die verschiedenen Funktionen der Debugger-Kommandos ist es wichtig, auf welche Weise der Debugger aktiviert wurde und welche Abläufe dabei unterbrochen wurden. Um dies zu unterscheiden, wird der Begriff ”Kontext” gebraucht. Kontext hat Einfluß auf die Definition bestimmter Kommando-Funktionen oder deren grundsätzliche Verwendbarkeit.

Es wird zwischen zwei Arten von Kontext unterschieden:

1. Escape-Kontext**:**
   Wenn der Debugger zuletzt durch eine Taste am Terminal gestartet wurde (Taskstart durch unerwartete Eingabe), dann gelten die Regeln für Escape-Kontext. Das heißt in erster Linie, daß keine Register einer Task verändert werden können. Die Kommandos für eine Task-Register-Manipulation stehen nicht zur Verfügung und der Debugger läuft mit eigener Task-ID und eigener Priorität. Der Zeichencode für Taskstart durch unerwartete Eingabe ist konfigurierbar.
2. Breakpoint-Kontext**:**
   Dieser ist gültig, wenn der Debugger durch einen Breakpoint aktiviert wurde (INT3- oder Debugregister-Breakpoint). Wenn in diesem Fall eine Task auf einen Haltepunkt gelaufen ist, wird sie angehalten und der Debugger gestartet. Im Breakpoint-Kontext können die Register (d.h. die CPU-Register zur Zeit der Unterbrechung) der angehaltenen Task angezeigt und verändert werden. Mit GO oder EXITK wird die Steuerung an die unterbrochene Task zurückgegeben. EXITK löscht zusätzlich alle gesetzten Breakpoints.

   Die Priorität des Debuggers im Breakpoint-Kontext beträgt immer 255.

Task-Bearbeitung

Der Debugger kann eine Task-Bearbeitung verbieten, d.h. mit Ausnahme des Debuggers wird keine weitere Task ausgeführt. Dies kann durch das INHIB-Kommando oder generell beim Start des Debuggers geschehen (wird bei der Konfigurierung festgelegt). Alle Task-Zustände bleiben während der abgeschalteten Task-Bearbeitung erhalten (Zustand eingefroren), bis die Task-Bearbeitung wieder freigegeben ist (Debugger-Beendigung oder INHIB). Bereits eingeleitete SVCs einer Task werden beendet, wenn zu deren Abarbeitung keine Task-Bearbeitung notwendig ist. So werden beispielsweise `RmIO`-Aufrufe einer Task beendet; die aufrufende Task befindet sich anschließend im Zustand BEREIT. EXIT, EXITK und GO beenden den Debugger, wodurch die Blockade der Task-Bearbeitung wieder aufgehoben wird.

Debugger-Start

Der Debugger meldet sich bei jedem Start z.B. mit:

```
RMOS3 DYNAMIC DEBUGGER, Vm.n
```

Hinweis

Wenn Sie mehrere Low-Level Debugger für die gleiche Task starten, kann es u. U. zu inkonsistenten Daten in den internen Registern der Task kommen. Bitte starten Sie **nur einen** Low-Level Debugger für eine Task, um diese Inkonsistenzen beim Debuggen zu vermeiden.

Initialisierung der Konsole

Wenn das Programm vom Debugger geladen wird, nimmt bereits der Borland-Start-Up-Code eines Testprogramms die Initialisierung der Systemkonsole vor.

Um die Konsole trotzdem umleiten zu können, muß das Pragma “startup” verwendet werden. In das Testprogramm muß eine Funktion analog zu folgendem Beispiel eingebaut werden:

```
void debug_init (void);
void debug_init (void)
{
   ...
   xinitt(...); // Device- und Unit-IDs mit RmGetEntry ermitteln
}
#pragma startup debug_init 0
```

Breakpoints in Codebereichen mit Scheduler-Sperre

Wenn in einem kritischen Code-Abschnitt, der mit den Aufrufen *RmDisableScheduler / RmEnableScheduler* gegen Taskwechsel geschützt ist, ein Breakpoint gesetzt wird, so wird beim Erreichen des Breakpoints die Sperre aufgehoben (wenn die Task auf diesen Breakpoint läuft, so wird sie vom Debugger blokkiert, und eine blockierte Task darf den Scheduler nicht sperren).

Nicht gemeldete Breakpoints

Wenn das Testprogramm durch einen Daten-Breakpoint unterbrochen wird, an einer Stelle, an der bereits ein Execution-Breakpoint steht, dann meldet der Low-Level Debugger nur den Daten-Breakpoint (z.B. Organon XDB Tracepoint). Der Execution-Breakpoint wird beim Fortsetzen des Testprogramms nicht mehr gemeldet.

6.3 Allgemeine Syntax-Regeln

Der Debugger meldet sich nach dem Start mit der Eingabeaufforderung:

```
DEBUG_T>
```

und erwartet folgendes Eingabeformat:

```
[<wiederholfaktor><leer>]<Kommando>[;[<wiederholfak-
tor><leer>]<Kommando>]...<return>
```

wobei

```
<Kommando> =<Kommandowort><DEL><Arg1><DEL><Arg2>...<Return>
```

Der Wiederholfaktor gibt an, wie oft das Kommando ausgeführt werden soll. 0 bedeutet, daß das Kommando beliebig oft ausgeführt wird. Dieser Parameter ist optional.

Das Kommandowort besteht aus mehreren Zeichen und bestimmt die gewünschte Funktion. Die Anzahl der Argumente ist kommandoabhängig.

Die Kommandowörter des Debuggers sind in einer Tabelle zusammengefaßt. Die CPU-Registernamen `<cpureg>` sind nur im Breakpoint-Kontext erlaubt. Alle Kommandowörter können auch in abgekürzter Form eingegeben werden, vorausgesetzt, daß die Abkürzung eindeutig unterscheidbar ist. So kann BREAKS mit BRE, BREA oder BREAK abgekürzt werden, das START-Kommando mit STAR, aber nicht mit ST, da diese Abkürzung nicht von STACK zu unterscheiden ist. Die Abkürzung muß mindestens 3 Buchstaben enthalten (Ausnahme EX für EXIT).

Für die Kommandowörter können Groß-und Kleinbuchstaben eingegeben werden. Der Debugger beginnt mit der Auswertung einer Eingabezeile erst nach der Eingabe von CR (Carriage Return).

Der Debugger fordert selbständig die Parameter an, wenn diese bei der Kommandoeingabe nicht angegeben werden. Bei Eingabe eines Kommandos ohne zugehörige Parameter gibt der Debugger eine Meldung aus und wartet auf die Eingabe der Parameter (interaktive Kommandoeingabe).

Bei einer Kommandozeile, in der mehrere numerische Parameter stehen, müssen einzelne Parameter, die ein Vorzeichen enthalten, z.B. -1, mit runden Klammern eingegeben werden (-1). Andernfalls wird das Vorzeichen als Operator behandelt und es erfolgt keine Trennung zum vorhergehenden Parameter.

Eine Eingabezeile kann mehr als ein Kommando enthalten. Mehrere Kommandos können nacheinander - durch Semikolon getrennt - bis zur Begrenzung durch die Kommandozeilenlänge in einer Zeile eingegeben werden. Vor dem Kommandonamen kann ein Wiederholungszähler stehen. Der Zählwert gibt an, wie oft das nachfolgende Kommando ausgeführt werden soll. Beim Wert Null für den Wiederholungszähler wird das Kommando ständig ausgeführt und kann nur durch CTRL-C abgebrochen werden.

Zahlenwerte werden bei der Eingabe standardmäßig dezimal oder hexadezimal interpretiert. Mit dem BASE-Kommando wird die standardmäßige Zahlenbasis eingestellt. Mit BASE=10 werden die Eingaben dezimal, mit BASE=16 hexadezimal interpretiert. Einzelne Werte können jedoch mit vorangestellten Zeichen (Präfix) in hexadezimaler, dezimaler, oktaler, dualer Zahlenbasis oder als ASCII-Zeichen eingegeben werden.

Präfixe für die Zahlenbasis:	0x	hexadezimal
	0n	dezimal
	0o	oktal
	0m	dual
	' '	ASCII

Ein ASCII-Wert wird angenommen, wenn ein oder mehrere Zeichen von Hochkommata (') eingeschlossen sind. Bei zwei Zeichen entspricht das linke Zeichen dem höherwertigen Byte des Wortes.

Die folgenden Zahlendarstellungen in einer Zeile sind gleichbedeutend:

0n09	0x09	0o011	0m00001001	(nicht abdruckbares Zeichen)
0n14	0x0E	0o016	0m00001110	(nicht abdruckbares Zeichen)
0n65	0x41	0o101	0m01000001	'A'

Gleitpunktzahlen müssen mit Dezimalpunkt oder mit Exponenten angegeben werden. Bei Real-Zahlen, die mit DISPLAY zur Anzeige gebracht werden, sind die Ziffern der Exponentendarstellung als dezimal zu interpretieren.

Beipiel für Gleitpunktzahlen:

```
CALC 3.141592654E+5 * 2.718281828E-3

+8.53973422234649D+002
```

Werden bei der Kommandoeingabe die Syntaxregeln nicht eingehalten, antwortet der Debugger mit seiner allgemeinen Fehlermeldung:

`Syntax error!`

und verlangt darauf eine neue Eingabe.

In den folgenden Abschnitten werden `Debugger-Ausgaben in anderer Schriftart` wiedergegeben. Ferner bedeutet:

xx	momentaner Speicherinhalt
CR	Kommando-Abschluß mit Carriage Return.

Die detaillierte Kommando-Syntax finden Sie im Referenzteil, Anhang C.

6.4 Kurzbeschreibung der Debugger-Kommandos

Übersicht

Die Kommandos des Debuggers können folgenden Gruppen zugeordnet werden:

Betriebsarten-Umschaltung

monitor	Debugger in den Monitor-Modus schalten
task	Debugger in den Task-Modus schalten

Tasks

start	Task starten, die im Zustand DORMANT ist
halt	Task anhalten
cont	Angehaltene Task freigeben
query	Task-Status abfragen
tcb	Daten vom TCB einer Task anzeigen
tcd	Daten vom TCD einer Task anzeigen
inhib	Task-Bearbeitung erlauben/sperren
loadtask	Task nachladen
freetask	Freigeben einer dynamisch geladenen Task
call	Ausführen eines Programms

Breakpoints

set	Breakpoint(s) setzen
breaks	Breakpoint(s) auflisten
kill	Breakpoint(s) löschen
qualify	Breakpoint parametrieren
switch	Breakpoint-Kontext wechseln

Speicher und Register

display	Speicherinhalt ausgeben
change	Speicherbytes anzeigen/verändern
fill	Speicherblock mit Byte-Werten füllen
regs	Alle Registerinhalte anzeigen
cpureg	Registerinhalt prüfen/verändern
stack	Stack anzeigen
asm	Speicherinhalt disassemblieren

Steuerungsübergabe

go	Unterbrochene Task freigeben und Debugger beenden
step	Einzelschritt ausführen
exitk	Breakpoints löschen und unterbrochene Task freigeben
exit	Debugger beenden

Ein-/Ausgabe

in	E/A-Port einlesen
out	E/A-Port ausgeben

System

svc	Systemaufruf generieren
dir	Einträge des Betriebsmittelkatalogs anzeigen
report	Resource Report
lines	Seitenweise Ausgabe ein/ausschalten

Numerik

help	Hilfe-Kommando
evaluate	Integer-Ausdruck berechnen
calculate	Gleitpunkt-Ausdruck berechnen
base	Zahlenbasis einstellen

Ladbare Treiber

7

In diesem Kapitel

In diesem Kapitel erfahren Sie:

- Welche ladbaren Treiber Sie einsetzen können
- Wie Sie einen Treiber laden und Treiber-Units erzeugen

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
7.1	Was Sie über ladbare Treiber wissen sollten	7-1
7.2	Laden eines Treibers	7-3

7.1 Was Sie über ladbare Treiber wissen sollten

Was ist ein Treiber?

Ein-/Ausgabeoperationen werden von M7 RMOS32 über Gerätetreiber, kurz Treiber oder Device genannt, und deren Untereinheiten, den Units, abgewickelt. Ein Gerätetreiber ist ein Software-Modul, das ein oder mehrere Peripheriegeräte gleichen Typs, bzw. deren Controller, steuert und verwaltet. Peripheriegerät gleichen Typs sind zum Beispiel mehrere Terminals, die über V.24-Schnittstellen angesteuert werden.

Wenn im folgenden von Geräten oder Units gesprochen wird, sind immer die von einem Treiber bedienten Controller und die Peripheriegeräte, die an die Controller angeschlossen sind, gemeint. Geräte können Ein-/ Ausgabegeräte oder Speicher sein.

Welche ladbaren Treiber gibt es?

Ladbare Treiber können im Gegensatz zu den fest eingebundenen Treibern bei Bedarf eingebunden werden, um bestimmte Peripheriegeräte zu steuern. Ab der Version V2.0 der Systemsoftware für M7-300/400 stehen folgende ladbaren Treiber zur Verfügung:

- 3964-Treiber
- ser8250-Treiber

Die Treiber bieten eine einheitliche Schnittstelle, über welche die M7 RMOS32-Tasks mit den Treibern kommunizieren können, um einen Treiber zu laden oder auf die Geräte zuzugreifen.

3964-Treiber

Character-Device-Treiber für Zugriffe auf die serielle Schnittstelle mit dem Protokoll 3964. Der Treiber unterstützt sowohl das 3964 als auch das 3964R Protokoll. Beide sind sichere Protokolle, bei denen die Daten byteweise, im asynchronen Halbduplex-Betrieb, transparent zum Kommunikationspartner übertragen bzw. von diesem empfangen werden.

Der Unterschied zwischen 3964 und 3964R besteht darin, daß bei 3964R nach Verbindungsabbau, der Sender dem Empfänger ein Blockprüfzeichen sendet. Das Blockprüfzeichen ist eine XOR-Verknüpfung über alle gesendeten Datenbytes.

SER8250-Treiber

Character-Device-Treiber für einfache Zugriffe auf die serielle Schnittstelle. Der Treiber unterstützt sowohl die bytewise Übertragung als auch die Übertragung von Zeichenketten.

Hardware

Die Treiber können folgende Hardware-Schnittstellen (Units) ansprechen:

- die COM-Schnittstelle der CPU 388-4 oder der FM 356-4
- die COM-Schnittstellen des Schnittstellenmoduls IF 961-COM
- sonstige serielle Schnittstellen, die auf dem UART-Baustein 8250 basieren, z. B. auf kurzen AT-Karten.

7.2 Laden eines Treibers

Einführung

Bevor Sie einen ladbaren Treiber benutzen können, muß der Treiber geladen und eine oder mehrere Units erzeugt werden. Dafür gibt es folgende Möglichkeiten:

- beim Systemhochlauf in der Konfigurationsdatei RMOS.INI
- im laufenden Betrieb:
  - im Kommandointerpreter CLI
  - im Anwenderprogramm

Laden eines Treibers beim Systemhochlauf

Um einen Treiber beim Systemhochlauf zu laden, rufen Sie in der Konfigurationsdatei RMOS.INI, Sektion RMOS das Kommado DEVICE auf:

```
DEVICE = <pfad name>  <unit name> [<unit parameter>]
```

Der Treiber wird geladen und in den Betriebsmittelkatalog eingetragen. Optional kann auch eine Unit erzeugt werden. Wird nur der Name der Unit angegeben, so wird eine Unit mit den voreingestellten Parametern erzeugt.

Um anschließend weitere Units dieses Treibers zu erzeugen, rufen Sie in der Konfigurationsdatei RMOS.INI das Kommado DEVICE noch einmal oder mehrmals auf:

```
DEVICE = <katalog name> <unit name> [<unit parameter>]
```

Wird nur der Name der Unit angegeben, so wird eine Unit mit den voreingestellten Parametern erzeugt.

Laden eines Treibers über CLI

Ein ladbarer Treiber kann auch im laufenden Betrieb über den Kommandointerpreter CLI geladen werden. Dafür wird das Kommando DEVICE benutzt. Das Vorgehen zum Laden des Treibers und zum Erzeugen der Units sowie die Kommandosyntax sind wie beim Eintrag in die Datei RMOS.INI, siehe oben “Laden eines Treibers beim Systemhochlauf”.

Die Aufrufe des Kommandos `DEVICE` in der Datei rmos.ini und im CLI unterscheiden sich durch folgendes: Hinter dem Kommando `DEVICE` kommt:

- ein Leerzeichen – im CLI
- ein “=”–Zeichen – in der rmos.ini

Laden eines Treibers aus dem Anwenderprogramm

Zum Laden eines Treibers und Erzeugen von Units im Anwenderprogramm steht der Funktionsaufruf **RmLoadDevice** des RMOS-API zur Verfügung. Die Beschreibung des Funktionsaufrufs und dessen Anwendung finden Sie im Referenzhandbuch bzw. im Kapitel “Ladbare Treiber” des Programmierhandbuchs.

Entfernen eines Treibers

Geladene Treiber können nicht aus dem laufenden Betriebssystem entfernt werden.

Laden des 3964-Treibers

Nachfolgend finden Sie Beispiele zum Laden des 3964-Treibers.

In der Datei rmos.ini

- 3964-Treiber in der Datei rmos.ini laden ohne eine Unit zu erzeugen:

  ```
  DEVICE = \M7RMOS32\3964.DRV
  ```

- 3964-Treiber laden und Unit für COM1 mit Voreinstellung erzeugen.

  ```
  DEVICE = \M7RMOS32\3964.DRV COM1 IRQ:4 BASE:0x3F8
  MODE:19200–N–8–1 PROT:1–1
  ```

  Der Treiber wird geladen und eine Unit namens `COM1` wird mit folgenden treiberspezifischen Parametern erzeugt: 19 200 bit/s, kein Paritybit, 8 Datenbits, 1 Stopbit, Protokoll 3964R, Master.

Im CLI:

- Treiber-Unit für 3964 an COM2 erzeugen:

  ```
  DEVICE 3964 3964_COM2 IRQ:3 BASE:0x2F8 MODE:19200 –N–8–1
  PROT:1–1
  ```

Laden des SER8250-Treibers

Nachfolgend finden Sie Beispiele zum Laden des SER8250-Treibers.

In der Datei rmos.ini

- SER8250-Treiber laden ohne eine Unit zu erzeugen:

  ```
  DEVICE = \M7RMOS32\SER8250.DRV
  ```

- SER8250-Treiber laden und Unit für COM1 mit Voreinstellung erzeugen.

```
DEVICE = \M7RMOS32\SER8250.DRV COM1 IRQ:4 BASE:0x3F8
MODE:19200–N–8–1
```

Der Treiber wird geladen und eine Unit namens `COM1` wird mit folgenden treiberspezifischen Parametern erzeugt: 19 200 bit/s, kein Paritybit, 8 Datenbits, 1 Stopbit

Im CLI:

- Treiber-Unit für SER8250 an COM2 erzeugen:

```
DEVICE SER8250 COM2 IRQ:3 BASE:0x2F8 MODE: 9600−N−8−1
```

Fehlerhafte Initialisierung von ladbaren Treiber-Units

Wenn Sie das Kommando DEVICE mit falschen Unit-Parametern absetzen, geschieht folgendes: Die Unit wird erzeugt und mit falschen oder mit keinen Werten initialisiert. Ein erneutes DEVICE-Kommando mit gleichem Unitnamen und neuen Unit-Parametern ist nicht möglich.

Um die Unit mit den richtigen Werten zu initialisieren, gehen Sie folgendermaßen vor:

- In RMOS.INI: Korrigieren Sie den DEVICE-Eintrag in der Datei RMOS.INI und booten Sie den M7-300/400 erneut.
- Im CLI:
  - Booten Sie den M7-300/400.
  - Rufen Sie das DEVICE-Kommando erneut auf, um die Unit neu zu erzeugen.

Betriebssysteme und Leistungsmerkmale A

In diesem Kapitel

In diesem Kapitel erfahren Sie:

- Die Leistungsmerkmale der M7-300/400 CPUs und FMs
- Wie die einzelnen Betriebssystemvarianten konfiguriert sind
- Wie der Hauptspeicher belegt ist
- Welche Komponenten von M7 RMOS32-DOS zur Verfügung stehen

Lesehinweis

Lesen Sie dieses Kapitel nur, wenn Sie Ihr vorkonfiguriertes Betriebssystem ändern müssen.

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
A.1	Leistungsmerkmale der Zentralbaugruppen und Applikationsbaugruppen	A-2
A.2	M7 RMOS32 - Konfigurierung	A-5
A.3	Belegung des Hauptspeichers	A-9
A.4	M7 RMOS32 für MS-DOS - Konfigurierung	A-10
A.5	Komponenten von M7 RMOS32-DOS	A-15
A.6	Besonderheiten unter M7 RMOS32-DOS	A-17

A.1 Leistungsmerkmale der Zentralbaugruppen und Applikationsbaugruppen

Übersicht

Leistungsmerkmal	CPU 388-4 FM 356-4	CPU 488-3 CPU 486-3 FM 456-4
Größe des Prozeßabbildes, jeweils der Eingänge und Ausgänge	256 Byte	512 Byte
Peripherieadressraum, Digitalein/ausgänge	8 KByte	
Peripherieadressraum, Analogein/ausgänge	8 KByte	
Merker	8*64 KBit (8*65535) von M 0.0 bis M 65535.7	
Zähler	0	
Timer	0	
Taktmerker	0	
Betriebsstundenzähler	0	
Bausteine: OBs FBs FCs DBs SFBs SFCs	 0 0 0 128*) 0 0	
Kommunikation über MPI • Übertragungsgeschwindigkeit • Anzahl Teilnehmer – ohne Repeater – mit Repeater • PDU-Größe	 187,5 kbit/s und 19,2 kbit/s**) max. 32 max. 127 max. 960 Byte	
PROFIBUS-DP-Kommunikation über IF 964-DP • Übertragungsgeschwindigkeit • Anzahl Teilnehmer	ja (s.Tabelle auf Seite 1-6) 9600 bit/s bis 12 Mbit/s 96	ja 9600 bit/s bis 12 Mbit/s 96
Industrial Ethernet (TCP/IP)-Kommunikation über CP1401 • Übertragungsgeschwindigkeit • Anzahl Sockets	ja (s.Tabelle auf Seite 1-6) 10 Mbit/s 44	ja 10 Mbit/s 44

*) Die Nummern von 0 bis 0xFFFF sind möglich, es wird jedoch empfohlen maximal 128 DBs einzusetzen.

**) Ab Version V4.0 wird auch die Übertragungsrate von 19,2 kBaud uterstützt (siehe unten)

S7-Verbindungenüber MPI-, PROFIBUS-DP und Industrial Ethernet (TCP/IP)

CPU 488-3, CPU 486-3:

Auf einer CPU der M7-400 Reihe können maximal 64 Verbindungen aufgebaut werden, die auf die folgenden Ressourcen verteilt werden können:

- bis zu 44 Verbindungen über den MPI-Anschluß,
- bis zu 56 Verbindungen über den K-Bus,
- bis zu 16 Verbindungen über Profibus-DP (IF 964-DP).
- bis zu 16 Verbindungen über Industrial Ethernet (CP 1401).

Von den 64 Verbindungen werden immer 4 Verbindungen für das Remote Filesystem und 4 Verbindungen für das Remote Terminal und den Organon Debugger reserviert. Zusätzlich werden jeweils eine PG-Verbindung und eine OP-Verbindung pro gestecktem Schnittstellenmodul und MPI reserviert. Die restlichen Verbindungen stehen für die Projektierung zur Verfügung.

FM 456-4:

Auf einer FM 456-4 können folgende Anzahlen von Verbindungen aufgebaut werden:

- bis zu 12 Verbindungen über den K-Bus ohne Schnittstellenmodule IF 964-DP und CP 1401
- zuzüglich bis zu 16 Verbindungen über Profibus-DP mit IF 964-DP
- zuzüglich bis zu 16 Verbindungen über Industrial Ethernet mit CP 1401

Z.B.: Eine FM 456-4 mit IF 964-DP und CP 1401 stellt maximal 44 Verbindungen zur Verfügung. Davon wird immer eine Verbindung für das Remote Filesystem und eine Verbindung für das Remote Terminal oder den Organon Debugger reserviert. Zusätzlich werden jeweils eine PG-Verbindung und eine OP-Verbindung pro gestecktem Schnittstellenmodul reserviert. Die restlichen Verbindungen stehen für die Projektierung zur Verfügung.

CPU 388-4, FM 356-4:

In einem Automatisierungssystem M7-300 können maximal 64 Verbindungen aufgebaut werden, die auf die folgenden Ressourcen verteilt werden können:

- bis zu 44 Verbindungen über den MPI-Anschluß,
- bis zu 16 Verbindungen über Profibus-DP (IF 964-DP).
- bis zu 16 Verbindungen über Industrial Ethernet (CP 1401).

Von den 64 Verbindungen werden immer 4 Verbindungen für das Remote Filesystem und 4 Verbindungen für das Remote Terminal und den Organon Debugger reserviert. Zusätzlich werden jeweils eine PG-Verbindung und eine OP-Verbindung pro gestecktem Schnittstellenmodul und MPI reserviert. Die restlichen Verbindungen stehen für die Projektierung zur Verfügung.

Hinweis

Nichtprojektierte Verbindungen sowie subnetzübergreifende Verbindungen, sind in der Ressourcenkontrolle durch STEP 7 z.Z. nicht berücksichtigt. Diese müssen jeweils von den restlichen, für die Projektierung zur Verfügung stehenden Verbindungen, abgezogen werden.

Remanente Daten (SRAM-Bereich)

Im SRAM(Static RAM)-Bereich liegen remanente Prozeßdaten (Datenbausteine und Merker), die auch nach einem Netzausfall verfügbar sind. Im Parametrierungsregister ”Remanenz” der M7-300/400 CPU und FM können Sie:

- bis zu 32 Datenbausteine und
- bis zu 256 Merker

als remanente Datenbereiche (im SRAM) projektieren.

Um einen Datenbaustein als remanent zu projektieren, geben Sie im Feld ”Anzahl Bytes” einen Wert $\neq 0$ ein. Damit ist der **gesamte** Datenbaustein remanent. Die Summe der Bytes aller remanenten Datenbausteine darf nicht 56 KByte überschreiten.

Uhrzeitsynchronisation

Ab STEP 7 V4 kann der SIMATIC M7 an der Uhrzeitsynchronisation im SIMATIC Kommunikationsnetz als Uhrenslave teilnehmen. Die Uhrzeitsynchronisation erfolgt immer in einem Kommunikationssegment, d.h. beim M7 innerhalb einer Station am K-Bus oder am P-Bus, sonst im MPI-Subnetz. In jedem Segment kann ein Uhrenmaster parametriert werden, in der Regel eine S7-300/400 CPU. Der Uhrenmaster sendet die aktuelle Uhrzeit in einem bestimmten Synchronisationsintervall. Die anderen Teilnehmer am Segment sind entweder Uhrenslaves oder verhalten sich neutral.

Die Tabelle A-1 zeigt die Synchronisationsarten, die für M7-Baugruppen im Register-Dialog “Diagnose / Uhr” einstellbar sind. Ein “–” bedeutet, daß keine Uhrzeitsynchronisation parametriert werden kann.

Tabelle A-1 SIMATIC M7 Uhrzeitsynchronisation

	Synchronisationsart	
Baugruppe	**Im AS (stationsintern)**	**Im MPI-Subnetz**
CPU 388-4	–	Uhrenslave
FM 356-4	Uhrenslave (über P-Bus)	–
CPU 488-3 CPU 486-3	Uhrenslave (über K-Bus, im CR2-Baugruppenträger)	Uhrenslave
FM 456-4	Uhrenslave (über K-Bus)	–

Korrekturfaktor

Die Abweichung der baugruppenspezifischen Uhrzeit von der Absolutzeit kann zusätzlich über einen Korrekturfaktor minimiert werden. Der Korrekturfaktor kann von –10000 bis +10000 ms/Tag eingestellt werden, Voreinstellung ist 0. Die Korrektur wird nicht auf einmal, sondern kontinuierlich über den Tag verteilt durchgeführt. Beim M7 bezieht sich der Korrekturfaktor auf die Softwareuhr. Für Netz-Aus-Zeiten erfolgt keine Korrektur.

FM 356-4 mit Lokalbussegment

Eine FM 356-4 mit Lokalbussegment überträgt im BZ RUN die Uhrzeittelegramme der CPU an die Baugruppen im lokalen Segment.

19,2 kbit/s an MPI

Die 19,2 kbit/s benötigen Sie nur, wenn eine Baugruppe am MPI-Subnetz angeschlossen wird, welche nur diese Übertragungsgeschwindigkeit unterstützt, z.B. eine CPU 214.

Hinweis

Bei 19,2 kbit/s für Kommunikation mit S7-200 sind **maximal 8 Teilnehmer** (CPU, PG/OP, FM/CP mit eigener MPI-Adresse) in einem Subnetz erlaubt.

Lesen Sie das *Systemhandbuch S7-200* für weitere Informationen!

Beachten Sie, daß Sie erst mit STEP 7 ab V 4.02 die 19,2 kbit/s einstellen können.

A.2 M7 RMOS32 - Konfigurierung

Belegung der Geräteeinheiten

A: Disketten-Laufwerk	Die Parameter werden dem CMOS entnommen
C: Festplatten-Laufwerk(e)	Die Partitionen der ersten Festplatte werden bei Systemstart automatisch integriert. Es werden die DOS-kompatiblen Partitionierungen erkannt. Von C ab werden die Partitionen fortlaufend mit Buchstaben bezeichnet: C: Partition 1 D: Partition 2 .....
M0:	Memory Card
M1	Onboard Silicon Disk (OSD)

EGA/VGA-Konfiguration

Zur optischen Unterscheidung wurden in der vorkonfigurierten Software folgende Farben vergeben:

RMOS-Konsole 0:	Unit 0: weiß
RMOS-Konsole 1	Unit 1: rot
RMOS-Konsole 2	Unit 2: blau
RMOS-Konsole 3	Unit 3: grün

Durch Betätigung von <Ctrl>+<D> kann auf den Ausgabegeräten (Units) ein Debugger aktiviert werden, durch <Ctrl>+<R> der Kommandozeileninterpreter CLI.

Systemkonsole

Die Systemkonsole ist das Ausgabegerät für RMOS-Systemmeldungen. Voreingestellt ist keine RMOS-Systemkonsole. Die Systemkonsole kann durch einen Eintrag in die Datei RMOS.INI auf ein Gerät umgelenkt werden, z.B. COM1/2 oder EGA/VGA1 bis EGA/VGA3.

Beim Systemstart wird automatisch die Systemkonsole aktiviert (wenn in der Datei RMOS.INI konfiguriert) und der CLI gestartet. Ein anderes Ausgabegerät wird durch Umschalten mit den Funktionstasten F2, F3 oder F4 angewählt.

Betriebssystemtakt

Der Betriebssystemtakt beträgt 1 ms, d. h. das kleinste Zeitintervall für die Aufrufe `RmPauseTask` und `RmRestartTask` beträgt 1 ms.

C-Laufzeitbibliothek CRUN

Die konfigurierte C-Laufzeitunterstützung entpricht dem ANSI-Standard Draft International Standard ISO/IEC DIS 9899 (herausgegeben 1990). Die Funktionen sind im Referenzhandbuch einzeln aufgeführt.

Systemtasks

Folgende RMOS-Systemtasks sind konfiguriert:

CLI_DPAT	CLI-Verteilertask	Bei Eingabe von <Ctrl>+<D> wird ein Debugger gestartet. Bei Eingabe von <Ctrl>+<R> wird die CLI-Logon-Task gestartet, die anschließend den CLI startet.
REP	Resource-Reporter Task	Ausgabe auf Unit 0. Aufruf aus dem Debugger mit START 2 oder durch das REP-Kommando.

DEB_0 DEB_1 DEB_2 DEB_3	Vier Debugger	Aktivierung mit <Ctrl>+<D>.
ERRLOG	Errorloggertask	
REMOTE	REMOTE-Task	Task zum Laden über serielle Schnittstelle
HSF_A HSF_B HSF_R	HSFS-Tasks für Diskettenlaufwerke	Je Speichermedium wird vom HSFS eine Task gestartet.
RMCONF	RMOS-Startup-Task	
BU_COUNT	Busy-Task	Mißt die Auslastung des Systems.
CLI_CLEANUP	CLI-Task für Hintergrundjobs	Bereinigt den Speicher bei Beendigung von Hintergrundjobs
CLI_JOB_0	Grundtask des CLI	CLI-Prompt
HSF_C	Task für Festplattenpartition C:	Je Festplattenpartition wird eine HSFS-Task gestartet.

Hinweis

Die Verteilertask prüft bei Eingabe von <Ctrl>+<D> nach, ob an der Konsole bereits ein Debugger gestartert wurde und startet diesen erneut. Ist an der Konsole noch kein Debugger vorhanden, so wird eine neue Debugger-Task erzeugt und gestartet.

Konfiguration des Datei-Verwaltungssystems

A: = Disketten-Laufwerk A
C: = Festplatten-Partition 1
(D: = Festplatten-Partition 2)
.
.
M0: = Memory Card.
M1: = OSD

M7 RMOS32 Treiber

Folgende M7 RMOS32-Treiber sind konfiguriert:

BYT-Treiber:	Device 0
Unit 0:	Systemkonsole (EGA/VGA-Unit 0), Tastatur (Eingabe)
Unit 1:	EGA/VGA-Unit 1
Unit 2:	EGA/VGA-Unit 2
Unit 3:	EGA/VGA-Unit 3
Unit 4:	serielle Schnittstelle (COM1, 19200 bit/s)
Unit 5:	serielle Schnittstelle (COM2, 19200 bit/s)
Unit 6:	Drucker (LPT1: Centronics)

FD0-Treiber, Disketten-Treiber: Device 1, Unit 0, Unit 1

HD0-Treiber, Festplatten-Treiber: Device 3, Unit 0, Unit 1

Memory-Card Treiber

OSD-Treiber

Bootvorgang

Der RMOS-Bootvorgang erfolgt nach DOS-Konvention, d.h. das im EPROM enthaltene BIOS lädt nach Systemreset den RMOS-Bootsektor (512 Bytes) von dem entsprechenden Bootmedium (Diskette, Festplatte, Memory-Card) in den Systemspeicher ab Adresse 0:0x7C00. Anschließend wird der Code des Bootsektors ausgeführt, der schließlich das als Speicherabbild vorliegende RMOS-System lädt und startet. Das RMOS-System muß sich dabei im Hauptverzeichnis des Bootmediums an erster Stelle befinden.

Bootmedium

Ein RMOS-Bootsektor wird auf Festplatten und Disketten mit dem Dienstprogramm RDISK eingerichtet. Auf Memory-Cards wird ein RMOS-Bootsektor mit dem Dienstprogramm FTLFORM.386 angelegt.

Dateiverwaltungssystem

M7 RMOS32 enthält das Dateiverwaltungssystem High-Speed File System (HSFS). Das Dateiverwaltungssystem kann Datenträger in einem DOS-kompatiblen Format verwalten.

Die Integration der Festplatten in das Dateiverwaltungssystem erfolgt beim Systemstart. In einer ersten Phase wird der HD0-Treiber initialisiert. Dazu werden die Daten der BIOS-ROM-Tabelle verwendet, auf die die Interrupt-Vektoren 41H und 46H zeigen. Nachdem der Treiber für Unit 0 (Laufwerk-C) und Unit 1 (Laufwerk-D) initialisiert wurde, wird in der zweiten Phase das Dateiverwaltungssystem initialisiert. Dabei wird jeder DOS-Partition ein ”Datenträgername” zugewiesen.

Die Partitionierung und Initialisierung der Festplatte wird mit dem Programm HDPART ausgeführt.

Watchdog

Zur Überwachung der Anwenderprogramme ist in den programmierbaren Baugruppen CPU/FM jeweils eine Zeitüberwachung vorhanden. Diese wird von der Systemsoftware zyklisch aufgezogen. Fällt aufgrund einer Störung das zyklische Aufziehen aus (z.B. durch Ressourcenmangel), so wird die Baugruppe nach Ablauf der Überwachungszeit zurückgesetzt.

A.3 Belegung des Hauptspeichers

Speicherbelegung

Der Hauptspeicher wird bei M7 RMOS32 und M7 RMOS32 mit MS-DOS wie folgt belegt:

Tabelle A-2 Speicheraufbau bei M7-300/400

Adresse	Inhalt
1MB bis 15MB 16MB bis Max. Speichergröße	Anwenderspeicherbereich
15MB bis 16MB	PROFIBUS-DP
E 8000H bis F FFFFH	BIOS
E 0000H bis E 7FFFH	frei (32K)
D 4000H bis D FFFFH	frei (48K)
D 0000H bis D 3FFFH	reserviert für CP 1401 (16 K)
C F000H bis C FFFFH	frei (4K)
C C000H bis C EFFFH	Memory Card und/oder OSD, ansonsten frei (12K)
C 8000H bis C BFFFH	Bei CPU 388-4, FM 356-4: SRAM Bei CPU 488/486-3, frei (16K) Bei FM 456-4 belegt (16K)
C 0000H bis C 7FFFH	Shadow VGA-BIOS (32K)
A 0000H bis B FFFFH	VGA (128K)
0 0000H bis 9 FFFFH	640 K Systembereich

Speicherbereiche für AT-Karten

AT-Karten, die in die Erweiterungsbaugruppe ATM 478 eingefügt werden, können folgende Speicherbereiche belegen:

Bereich	M7 RMOS32	M7 RMOS32 mit MS-DOS	
		ohne EMS	mit EMS
D 4000H bis E 7FFFH	80K	80K	16K[3)]
C 8000H bis C BFFFH[1)]	16K	16K	16K
C C000H bis C EFFFH[2)]	12K	12K	12K
C F000H bis C FFFFH	4K	4K	4K

1) Der Bereich ist nur bei CPU 488/486-3 verfügbar, bei CPU 388-4 und FM 356-4 ist der Bereich für SRAM belegt.

2) Der Bereich ist nur verfügbar, wenn keine Memory Card oder OSD vorhanden ist.

3) **W**enn die Treiber-Software unter MS-DOS erweiterten Speicher (EMS) benötigt, belegt der Speichermanager EMM386 64K im Bereich D0000H bis E7FFFH, weil er im EMS-Mode betrieben werden muß.

A.4 M7 RMOS32 für DOS - Konfigurierung

Massenspeicher

A: Disketten-Laufwerk

B: Memory Card-Laufwerk

C: Festplatten-Laufwerk oder OSD (wenn keine Festplatte vorhanden ist)

D: ... OSD, wenn auch eine Festplatte vorhanden ist, oder weitere Festplattenpartitionen

Alle Massenspeicher werden vom DOS-Dateiverwaltungssystem bedient.

Parallele Schnittstellen

Alle parallelen Schnittstellen werden von MS-DOS bedient.

Serielle Schnittstellen

COM1 - wird von MS-DOS bedient.

COM2 - wird vom BYT-Treiber des M7 RMOS32 bedient. COM2 ist vorgesehen für den Anschluβ eines Terminals (19200 Baud, 8 Datenbits, Parity off, 2 Stopbits, 3-Draht-Verbindung (TxD, RxD, GND) und im Stecker der M7-Seite DTR mit DSR, RTS mit CTS verbunden).
COM2 kann als Systemkonsole des M7 RMOS32 eingestellt werden, d.h. die Systemausgaben erfolgen über diese Schnittstelle.

Betriebssystemtakt

Der Betriebssystemtakt beträgt 1 ms, d.h. das kleinste Zeitintervall für die Aufrufe `RmPauseTask` und `RmRestartTask` beträgt 1 ms.

C-Laufzeitbibliothek CRUN

Die konfigurierte C-Laufzeitunterstützung CRUN entspricht dem ANSI-Standard Draft International Standard ISO/IEC DIS 9899 (herausgegeben 1990). Die Funktionen sind im Referenzhandbuch einzeln aufgeführt.

Systemtasks

Folgende RMOS-Systemtasks sind konfiguriert:

WINCOND_0 bis WINCOND_7	Virtuelle Konsole (Daemon)	
CLI_DPAT	CLI-Verteilertask	Bei Eingabe von <Ctrl>+<D> wird ein Debugger gestartet. Bei Eingabe von <Ctrl>+<R> wird die CLI-Logon-Task gestartet, die anschließend den CLI startet.
REP	Resource-Reporter Task	
DEB_0 DEB_1 DEB_2	Drei Debugger	Aktivierung mit <Ctrl>+<D>.
ERRLOG	Errorloggertask	
REMOTE	REMOTE-Task	
RMCONF	RMOS-Startup-Task	
DOS_HSFS	HSFS-Task für Dateiverwaltungssystem	
DOS_TASK	DOS-Task	Ist unter zweiten Namen DOS katalogisiert.
CLI_CLEANUP	CLI-Task für Hintergrundjobs	Bereinigt den Speicher bei Beendigung von Hintergrundjobs
BU_COUNT	Busy-Task	Mißt die Auslastung des Systems.

Treiber

Folgende RMOS-Treiber sind konfiguriert:

BYT-Treiber:	Device 0 Unit 0:	Systemkonsole (COM2)
VC-Treiber:	Device 1	
	Unit 0 . Unit 7	Units, die vom DOS-Programm RM3_CON.EXE auf Monitor und Tastatur abgebildet werden.

Betriebsmittel

10	statische Mailboxen
10	statische globale Eventflaggruppen
10	statische Semaphore
1	Speicherpool

Speicherpool

Es ist ein Speicherpool als HEAP konfiguriert. Die Größe des Pools hängt von dem für RMOS zur Verfügung gestellten Speicher ab.

Bootvorgang

DOS wird zuerst gebootet. Der Treiber RM3PMEM.SYS muβ an erster Stelle in der Datei CONFIG.SYS stehen. Der Treiber legt entsprechend seinen Aufrufparametern die Speicheraufteilung zwischen RMOS und DOS fest.

Wenn ein von RMOS unabhängiger Warmstart des DOS durchgeführt werden soll, muβ der Treiber RM3RESET direkt hinter RM3PMEM.SYS oder bei Verwendung eines Memory-Managers hinter dem Memory-Manager eingetragen sein. RM3RESET.SYS speichert das aktuelle DOS-Speicherabbild (ca. 800 Kbyte) in dem für RMOS reservierten Speicher. Die Ladeadresse, ab der das RMOS-System dann physikalisch geladen wird, wird entprechend nach oben verschoben.

Das speicherresidente DOS-Programm RM3_TSR.EXE wird vor dem Laden von RMOS aus der Datei AUTOEXEC.BAT gestartet.

M7 RMOS32 wird mit dem Ladeprogramm RUN_RM3.EXE geladen. Dieses Programm liest die RMOS-Systemdatei RM3DOS.LOC und lädt das System in den für M7 RMOS32 reservierten Speicher. Das laufende DOS-System wird dabei nicht überschrieben. Nach dem Laden wird M7 RMOS32 direkt gestartet. Das DOS-System bleibt zu diesem Zeitpunkt quasi eingefroren.

Die DOS-Task wird dann von der Initialisierungstask gestartet.

Die DOS-Task initialisiert die gesamte RMOS-DOS-Umgebung und reaktiviert das eingefrorene DOS-System. Aus Sicht von DOS wurde bisher nur das laufende Ladeprogramm RUN_RM3.EXE ausgeführt.

Speicheraufteilung

Nach dem Booten und Hochlaufen von RMOS-DOS wird der Speicher wie in Tabelle A-2 belegt.

Warmstart mit <Ctrl> <Alt> <Del>

Es ist möglich, einen DOS-Warmstart durchzuführen, ohne daβ das laufende RMOS-System unterbrochen wird. Der Warmstart wird durch die Tastenkombination <Ctrl>+<Alt>+<Del> oder von RMOS durch Start der DOS-Warmstarttask bzw. einen Funktionsaufruf einer anderen Task ausgelöst.

Der Warmstart setzt voraus, daβ der Treiber RM3RESET.SYS korrekt in der CONFIG.SYS installiert wurde.

Dateiverwaltungssystem

Unter RMOS-DOS werden Festplatte(n) und Diskettenlaufwerk(e) grundsätzlich von DOS verwaltet. Ein direkter Plattenzugriff von RMOS aus ist nicht möglich.

Das RMOS-DOS-Dateiverwaltungssystem ist eine Schnittstelle zwischen dem RMOS-Dateiverwaltungssystem (HSFS) und dem DOS-Dateiverwaltungssystem, mit der eine RMOS-Task auf die von DOS verwalteten Laufwerke zugreifen kann. Es wird in das HSFS über die Netzwerkschnittstelle als Netzwerk-Task eingebunden. Damit werden die Aufrufe von RMOS-Tasks an das Dateiverwaltungssystem zuerst vom HSFS entgegengenommen und dann an das DOS-Dateiverwaltungssystem weitergegeben.

Laufwerke

In der Betriebssystem-Konfiguration M7 RMOS32 mit MS-DOS sind 26 Laufwerke mit den Namen A bis Z definiert. Damit können Sie auf alle unter DOS konfigurierten Laufwerke zugreifen. Es spielt keine Rolle, ob diese Laufwerke unter DOS mit Netzwerksoftware realisiert oder lokal vorhanden sind.

Task-Verwaltung unter M7 RMOS32 mit MS-DOS

M7 RMOS32 kann MS-DOS-Programme unterbrechen:

Hinweis

MS-DOS ist ein Mono-Task Betriebssystem.

Dies bedeutet, daß ein MS-DOS-Programm jederzeit durch ein von M7 RMOS32 gestartetes Programm unterbrochen werden kann. Beispielsweise läuft kein anderes MS-DOS-Programm wenn die RMOS-Konsole aktiv ist, weil diese auch ein TSR-DOS-Programm ist.

Um dies zu vermeiden, verwenden Sie das Remote Terminal Interface (siehe Kapitel 4.1).

In M7 RMOS32 mit MS-DOS-Systemen erfolgt der Zugriff auf Massenspeicher und Ein-/Ausgabegeräte (VGA-Monitor und Tastatur) über das DOS-Dateisystem. Der Zeitbedarf für diese Zugriffe ist aber nicht deterministisch und kann dadurch für die Anwendungstask mit z.B. wichtigen Steuerungsaufgaben zu Wartezeiten führen, die der vergebenen Priorität nicht entsprechen.

Umgekehrt kann es vorkommen, daß M7 RMOS32-Aktivitäten von DOS-Befehlen unterbrochen werden:

Z.B. das Kommando **dir** unter MS-DOS hält die laufenden Dateisystem-Aktivitäten unter M7 RMOS32 an. Wird beispielsweise das Kommando **dir** während eines Downloads von Datenbausteinen auf den M7 aufgerufen, so wird der Download angehalten und erst nach Beenden des **dir** fortgesetzt.

Hinweis

Um unerwünschte Auswirkungen zu vermeiden, sollten Schreib- bzw. Lesevorgänge an das DOS-Dateisystem sowie Zugriffe auf Ein-/Ausgabegeräte einer eigenständigen niederprioren Task übergeben werden.

Schnittstellen zwischen M7 RMOS32 und MS-DOS

M7 RMOS32 mit MS- DOS stellt Funktionsaufrufe für die Kommunikation von einem Betriebssystem zum anderen zur Verfügung. Diese Schnittstellen sind wegen der Kompatibilität zu früheren M7-SYS-Versionen vorhanden und nicht für Neuentwicklungen vorgesehen.

A.5 Komponenten von M7 RMOS32-DOS

Folgende Programme sind in die Systemdateien AUTOEXEC.BAT und CONFIG.SYS per Voreinstellung eingetragen:

In AUTOEXEC.BAT:

- REMAP_A.EXE - Floppy A: unter MS-DOS wieder zugänglich machen
- RM3_CON.EXE - Treiber für virtuelle RMOS-Konsolen
- RM3_TSR.EXE - Schnittstelle zwischen RMOS und DOS
- RUN_RM3.EXE - zum Laden und Starten von RMOS

In CONFIG.SYS:

- RM3PMEM.SYS - für die Speicheraufteilung zwischen RMOS und DOS
- RM3RESET.SYS - für die Bearbeitung eines Warmstarts von DOS

Hinweis

Das Betriebssystem ist für Ihre M7-300/400 Hardware-Konfiguration vorkonfiguriert, es bedarf in der Regel keiner Änderungen in den Dateien AUTOEXEC.BAT und CONFIG.SYS.

Bitte ändern Sie, wenn erforderlich, nur die nachfolgend beschriebenen Einträge.

A.5.1 RM3PMEM.SYS

Funktion

RM3PMEM.SYS ist ein DOS-Gerätetreiber, der die Speicheraufteilung zwischen RMOS und DOS festlegt.

Syntax

DEVICE=<Verzeichnis>RM3PMEM.SYS RMOSSTART=nnnnn
(in der Datei CONFIG.SYS)

Schalter

Beschreibung

RMOSSTART = nnnnn	nnnnn gibt die physikalische Speicheradresse an, ab der der Speicher für RMOS verwendet wird. Der Wert wird hexadezimal eingegeben. Ein Wert kleiner als 110 000 ist unzulässig. Der Default-Wert ist konfigurationsabhängig, z.B. 380 000 bei einem Hauptspeicherausbau von 8 Mb.

Dieser Treiber reserviert den Speicherbereich ab der Adresse, die mit dem Parameter RMOSSTART angegeben wird, bis zum Ende des physikalisch vorhandenen Speicher für RMOS. DOS steht nur der Speicher bis zu dieser Adresse zur Verfügung. Die Einstellung kann nur in 4-Kbyte-Schritten vorgenommen werden.

Hinweis

Der für RMOS reservierte Speicherbereich muß mindestens 4,5 Mbyte groß sein.

A.5.2 RM3RESET.SYS

Funktion

RM3RESET.SYS ist ein DOS-Gerätetreiber, der für die Bearbeitung eines Warmstarts von DOS notwendig ist.

Syntax

(in der Datei CONFIG.SYS)

DEVICE=<Verzeichnis>RM3RESET.SYS

Beschreibung

Dieser Treiber wird nur benötigt, wenn während des laufenden Betriebs von RMOS ein Warmstart von DOS durchgeführt werden soll. RM3RESET.SYS speichert das aktuelle DOS-Memory-Image (ca. 800 Kbyte) am Anfang des für RMOS reservierten Speicherbereichs und verschiebt entsprechend die RMOS-Ladeadresse nach oben.

A.6 Besonderheiten unter M7 RMOS32-DOS

Programme, die unter RMOS-DOS nicht ablauffähig sind

Einige DOS-Programme können nicht unter RMOS-DOS ausgeführt werden. Es handelt sich um :

- Programme, die den Interruptcontroller neu programmieren.
- Programme, die Timer 0 oder Timer 1 neu programmieren. Dagegen ist der Zugriff auf dem Timer 2 zulässig.

Software, die einen der folgenden DOS-Extender benutzt, kann auf M7 RMOS32 mit MS-DOS nicht eingesetzt werden, wie z.B.:

- Phar Lap TNT DOS-Extender (z.B. unter DOS Visual C++, MASM V6.1, ACAD V12, Borland C V4.02, V4.5)
- DOS/4GW 32-Bit DOS-Extender von Rational Systems (enthalten z.B. bei Watcom C++ 9.5)
- DOSX aus Symantec C++ Professional 6.1

Die Benutzung des Editors von Quick-C unter MS-DOS kann sporadisch zum Absturz des M7 RMOS32-Betriebssystems führen.

Hinweis

M7 RMOS32 mit MS-DOS ist mit der Entwicklungsumgebung Quick-C **nicht freigegeben**.

Die Benutzung dieses Editors im laufenden Betrieb ist nicht erlaubt.

DOS-Debugger

Wenn ein Debugger in der DOS-Task verwendet wird, ist es nicht möglich, einen Befehl INT n im Single-Step-Mode auszuführen, wenn dieser Interrupt nicht von DOS sondern von RMOS-DOS behandelt wird. Dies gilt in jedem Fall für den EMS-Interrupt (67H) und den vom Programm RM3_TSR benutzten Interrupt (79H). Es wird empfohlen, einen Breakpoint nach dem Befehl INT n zu setzen anstatt einen Single Step zu benutzen.

RMOS-DOS-Dateiverwaltungssystem

Es bestehen einige Unterschiede zwischen dem RMOS-DOS-Dateiverwaltungssystem und dem original HSFS. Diese Unterschiede stammen aus grundsätzlichen Unterschieden zwischen dem DOS- und dem RMOS-Dateiverwaltungssystem.

Maximale Zahl offener Dateien

Die maximale Zahl von offenen Dateien unter dem RMOS-DOS-Dateiverwaltungssystem beträgt 64 und wird auβerdem durch die folgenden Variablen eingeschränkt:

- die Zahl, die in der DOS-Datei CONFIG.SYS mit ”FILES=nn” angegeben wird.
- die Gröβe der Job File Table (JFT) im speicherresidenten DOS-Programm RM3_TSR.EXE. Diese Tabelle hat 64 Einträge.

Jede der oben aufgeführten Variablen legt implizit die mögliche Anzahl offener Dateien fest. Der kleinste dieser Werte definiert die maximal erlaubte Anzahl von offenen Dateien in Ihrem System.

Lesen einer Verzeichnisdatei

Das HSFS läβt zu, daβ eine Verzeichnisdatei geöffnet und gelesen wird. Dies ist mit dem DOS-Dateiverwaltungssystem nicht möglich.

Das RMOS-DOS-Dateiverwaltungssystem läβt zu, daβ eine Verzeichnisdatei geöffnet wird, jedoch nicht, um aus ihr Daten zu lesen. Die dabei zurückgegebene Kanalnummer kann anschlieβend beim Öffnen von Dateien in diesem Verzeichnis verwendet werden.

32-Bit-Verzeichniseintrag

Beim HSFS wird der volle 32-Bit-Verzeichniseintrag zurückgegeben. Unter DOS kann die Verzeichnisdatei nicht gelesen werden. RMOS-DOS erzeugt deshalb einen Pseudo-Verzeichnis-Eintrag, in dem die folgenden Felder vorhanden sind :

- Dateiname,
- Dateiattribute,
- Datum und Uhrzeit,
- Dateigröβe.

Die erste Cluster-Nummer in der Datei wird in diesem Pseudo-Verzeichnis-Eintrag nicht eingetragen.

Verkehrte Konsolenzuordnung nach Warmstart von MS-DOS

Wenn nach einem Warmstart von MS-DOS die Zuordnung der RMOS-Konsolen nicht mehr stimmt, führen Sie bitte einen erneuten Warmstart von MS-DOS aus, um die korrekte Konsolenzuordnung wiederherzustellen.

Start-Meldung unter MS-DOS

Versuchen Sie eine RMOS-Applikation unter MS-DOS zu starten, erscheint die Fehlermeldung: ”This program must be run under Win32”.

Hinweis: Bitte starten Sie das Programm unter M7 RMOS32, z.B. im CLI.

Abbruch von DOS

Wenn Sie den Editor **edit** von MS-DOS benutzen und mit CTRL-ESC auf M7 RMOS32 schalten, kann es zum Absturz von MS-DOS kommen.

Abhilfe: Sie können das Problem vermeiden, indem Sie:

- **edit** beenden, bevor Sie zu M7 RMOS32 umschalten oder
- das Kommando DOSKEY in die Datei AUTOEXEC.BAT eintragen.

CLI–Kommandos

B

Kapitelübersicht

Kapitel	Kommando	Beschreibung	Seite
B.1	BYT8250	Parametrieren des BYT-Treibers für die serielle Schnittstelle (UART)	B-3
B.2	CANCEL	Kommandos abbrechen	B-5
B.3	CD	Verzeichnis wechseln oder aktuelles Verzeichnis anzeigen	B-6
B.4	CHGKBD	Tastaturlayout umschalten	B-7
B.5	COPY	Eine oder mehrere Dateien kopieren, wahlweise mit Unterverzeichnissen	B-8
B.6	CPRI	Task-Priorität eines CLI-Jobs ändern	B-10
B.7	DATE	Aktuelles Datum anzeigen oder ändern	B-11
B.8	DEL	Eine oder mehrere Dateien löschen	B-12
B.9	DEVICE	Treiber laden oder Treiber-Unit erzeugen	B-13
B.10	DIR	Inhaltsverzeichnis auflisten	B-16
B.11	DISMOUNT	Datenträger abmelden	B-17
B.12	ECHO	Anzeigen der Befehlszeile bei Batchdatei-Bearbeitung ein- oder ausschalten, oder eine Meldung ausgeben	B-18
B.13	ERROR	Errorlevel und seine Bedeutung anzeigen	B-19
B.14	EXIT	CLI-Task beenden	B-20
B.15	FORMAT	Diskette oder Festplatte formatierten	B-21
B.16	FTLFORM	Memory Card oder OSD formatieren	B-22
B.17	HELP	Kommandoübersicht oder Syntaxerklärung für einzelne Kommandos ausgeben	B-24
B.18	MD	Neues Verzeichnis anlegen	B-25
B.19	MOUNT	Datenträger anmelden oder Zustand aller Laufwerke anzeigen	B-26
B.20	NPX	Präfix zum Starten eines Programms, das den numerischen Coprozessor verwendet	B-27
B.21	PATH	Suchpfad für nachladbare CLI-Kommandos festlegen	B-28
B.22	PING	Prüfen, ob ein Netzwerkteilnehmer aktiv ist	B-30
B.23	PROMPT	Eingabeaufforderung des CLI ändern	B-31

Kapitel	Kommando	Beschreibung	Seite
B.24	RD	Leeres Unterverzeichnis löschen	B-32
B.25	RDISK	Bootsektor auf Festplatte einrichten	B-33
B.26	RENAME	Eine oder mehrere Dateien bzw. Verzeichnisse umbenennen	B-34
B.27	SCANDISK	Datenträger überprüfen und korrigieren	B-35
B.28	SESSION	Eine Vordergrundtask auf einem anderen Terminal starten	B-37
B.29	SET	Environment-String definieren	B-38
B.30	START	Einen Hintergrund-Job starten	B-39
B.31	SYSTAT	Aktive CLI-Jobs anzeigen	B-40
B.32	TIME	Die aktuelle Uhrzeit anzeigen oder ändern	B-41
B.33	VER	RMOS Systeminformationen ausgeben	B-42
B.34	CLI-Fehlermeldungen		B-43

Hinweis

Bei der Beschreibung der Kommandos wird jeweils angegeben, ob es sich um ein Inline- oder ein nachladbares Kommando handelt.

Für den Anwender ist es unwesentlich, ob es sich um ein Inline- oder ein nachladbares Kommando handelt. Der einzige auffällige Unterschied ist, daß die Inline-Kommandos nicht abgebrochen werden können und ohne Ladevorgang zur Verfügung stehen. Das heißt, sie sind auch ausführbar, wenn keine Massenspeicher existieren.

B.1 BYT8250

Funktion **Parametrieren des BYT–Treibers für die serielle Schnittstelle (UART)**

nachladbares Kommando

Syntax **byt8250 name port baud mode [irq entry [unstask key1 key2]]**

Parameter-name	**Bedeutung**
name	Name der Unit (BYT_COM1, BYT_COM2)
port	Portadresse des UART–Bausteins (hex) Für BYT_COM1 0x3F8, für BYT_COM2 0x2F8.
baud	Baudrate des UART–Bausteins (dezimal)
mode	Betriebsmodus des UART–Bausteins (hex) **Daten– bzw. Stopbits** 0x02 7 Datenbits 0x03 8 Datenbits 0x00 1 Stopbit 0x04 2 Stopbits **Parität** 0x00 keine Parität 0x18 gerade Parität 0x08 ungerade Parität **Emulation** 0x0000 SME–Emulation 0x1000 VT100–Emulation **Modus** 0x0100 Transparent Mode 0x0000 Terminal Mode Die jeweiligen Werte müssen mit ODER verknüpft werden. **Beispiel**: Der Hex–Wert 0x1103 bezeichnet folgenden Betriebsmodus: Der UART–Baustein arbeitet mit 8 Datenbits, 1 Stopbit, keine Parität, VT100–Emulation im Transparent Mode.
irq	Hardware–Interrupt des UART–Bausteins (dez) z.B. 4=IRQ4 Für BYT_COM1 4, für BYT_COM2 3.
entry	Katalogeintrag, unter dem die Adresse des Interrupt–Handlers hinterlegt ist. Für BYT–Treiber immer X_BYT_COM_INTR.

Parameter-name	Bedeutung
unstask	Name der Unsolicited Task (Verteilertask). Für BYT–Treiber immer CLI_DPAT. Die CLI_DPAT Verteilertask übernimmt je nach Tastenkombination (CTRL+R bzw. CTRL+D) den Start des CLI bzw. des Debuggers.
key1 key2	Tastenkombinationen zum Starten der Unsolicited Task. 1=CTRL+A,.., 26=CTRL+Z Für die Task CLI_DPAT immer 4 18. Durch die Tastenkombination CTRL+D wird der Debugger, durch die Kombination CTRL+R der CLI gestartet.

Beschreibung

Das Kommando `BYT8250` erlaubt eine Umparametrierung des UART–Bausteins für die serielle Schnittstelle.

Dabei bezeichnet *name* den Namen der Unit, *port* die Portadresse und *baud* die Baudrate des UART–Bausteins. *mode* gibt den Betriebsmodus des UART–Bausteins an.

Unter *irq* wird der Hardware–Interrupt des UART–Bausteins dezimal angegeben. *entry* gibt den Katalogeintrag für die Adresse des entsprechenden Interrupt–Handlers an.

unstask ist der Name der Unsolicited Task (Verteilertask). Sie erlaubt das Starten ablauffähiger Programme über die mit *key1 key2* angegebenen Tastenkombinationen.

Hinweis

Eine mit dem Kommando `BYT8250` eingerichtete Unit darf nicht als Systemkonsole verwendet werden.

Beispiel

```
byt8250 BYT_COMNEU 0x2F8 19200 0x1003 3 X_BYT_COM_INTR CLI_DPAT 4 18
```

Nach diesem Aufruf ist der UART–Baustein mit der Portadresse 0x2F8 (entspricht BYT_COM2) der Unit BYT_COMNEU zugeordnet. Eine vorher zugeordnete Unit ist dann zwar noch vorhanden, kann aber nicht mehr auf den UART–Baustein zugreifen.

Dieses Kommando können Sie auch in der Systemdatei RMOS.INI eintragen. Sie müssen dem Kommando nur RUN = voranstellen.

```
RUN = byt8250 BYTE_COMNEU 0x2F8 19200 0x1003 3 X_BYT_COM_INTR
              CLI_DPAT 4 18
```

Dadurch wird bei jedem Hochlauf des Systems die Unit automatisch umkonfiguriert.

B.2 CANCEL

Funktion Kommandos abbrechen

Inline-Kommando

Syntax cancel <job number>

Parametername	Bedeutung
job number	Job-Nummer, die beim Starten der Task vergeben wurde. Die Nummer wird mit dem Kommando SYSTAT angezeigt.

Beschreibung

Der mit *job number* spezifizierte Job wird abgebrochen. Vordergrund-Jobs können nicht abgebrochen werden. Besonderheiten bei Programmabbruch siehe auch Kapitel 15.4.3 im Benutzerhandbuch.

Falls eine Batchdatei als Hintergrund-Job ausgeführt wird, ist es möglich, daß der ursprünglich gestartete Job auf die Beendigung eines weiteren CLI-Jobs wartet. In diesem Falle gibt es die folgende zwei Möglichkeiten:

- Durch Angabe der Job-Nummer des ursprünglich gestarteten Hintergrund-Jobs werden automatisch beiden Jobs abgebrochen. (Zuerst der untergeordnete und anschließend der ursprüngliche).
- Durch Angabe der Job-Nummer des untergeordneten Jobs wird nur dieser Job abgebrochen. Der ursprüngliche Job läuft weiter und führt das nächste Kommando der Batchdatei aus.

  Die Job-Nummer des untergeordneten Jobs wird ebenfalls mit dem Kommando SYSTAT angezeigt.

Inline-Kommandos und Vordergrund-Jobs können mit `CANCEL` nicht abgebrochen werden.

Hinweis

Bei gesetzter Scheduling-Sperre (`RmDisableScheduler`) ist <Ctrl>+<C> nicht wirksam.

Beispiel

```
CANCEL 2
```

Der Hintergrund-Job Nummer 2 wird abgebrochen.

B.3 CD

Funktion Verzeichnis wechseln oder aktuelles Verzeichnis anzeigen

Inline-Kommando

Syntax cd [<*Verzeichnis*>]

Parametername	Bedeutung
Verzeichnis	Gültiges Verzeichnis in der Standard-Schreibweise mit oder ohne Laufwerksangabe.

Beschreibung

Mit `CD` wird das aktuelle Laufwerk und Verzeichnis festgelegt.

Das Verzeichnis kann relativ zum Hauptverzeichnis oder zum aktuellen Verzeichnis definiert werden. Das übergeordnete (Parent) Verzeichnis kann mit ”..” angegeben werden. Eine Eingabe ohne Parameter zeigt das aktuelle Verzeichnis an.

Jede Task hat ihr eigenes aktuelles Laufwerk und Verzeichnis, das sie mit der CRUN-Funktion `chdir` ändern kann. Beim Starten eines Jobs (im Vordergrund oder Hintergrund) wird das aktuelle Laufwerk und Verzeichnis gleich dem der CLI-Session gesetzt.

Beim Wechseln auf ein anderes Laufwerk wird dieses automatisch mit dem MOUNT angemeldet.

Im Gegensatz zu DOS gibt es unter dem CLI kein aktuelles Laufwerk sondern **nur ein** aktuelles Verzeichnis, das das Laufwerk beinhaltet. Die Angabe A: ist gleichbedeutend mit A:\ .

Beispiel

```
CD MY_DIR
CD ..\TEMP
CD \TOOLS
CD A:
CD A:\INC
```

B.4 CHGKBD

Funktion Tastaturlayout umschalten

nachladbares Kommando

Syntax chgkbd <type>

Parametername	Bedeutung
`type`	Angabe des Tastaturlayouts Es gibt 6 verschiedene Varianten: KBDUSE (**U**nited **S**tates **E**nglisch) KBDUKE (**U**nited **K**ingdom **E**nglisch) KBDGER (**Ger**man) KBDFRE (**Fre**nch) KBDITA (**Ita**lian) KBDSPA (**Spa**nish)

Beschreibung

Das Kommando `CHGKBD` schaltet die Tastatur auf die mit *`type`* angegebene Belegung um.

Die entsprechenden sechs Layoutdateien KBDxxx.SYS müssen sich im selben Verzeichnis wie das Kommando `CHGKBD` befinden.

Beispiel

```
CHGKBD KBDGER
```

Schaltet auf deutsche Tastaturbelegung um.

Wenn Sie während des Systemhochlaufs die Tastatur automatisch umschalten wollen, ist dies über folgenden Eintrag in der Datei RMOS.INI an beliebiger Stelle möglich.

```
RUN=C:\BIN\CHGKBD.386 KBDGER
```

Das Kommando `CHGKBD` und die Tastaturbelegungsdatei KBDGER.SYS müssen sich im Verzeichnis C:\BIN befinden.

B.5 COPY

Funktion **Eine oder mehrere Dateien kopieren, wahlweise mit Unterverzeichnissen**

nachladbares Kommando

Syntax **copy <old file spec> [<new file spec>] [/S] [/L]**

Parametername	Bedeutung
`old file spec`	Die Datei oder Dateien die zu kopieren sind. Beim Dateinamen und der Erweiterung können die Platzhalter * und ? verwendet werden.
`new file spec`	Das Verzeichnis oder der Dateiname für die neue(n) Datei(en). Beim Dateinamen und der Erweiterung können die Platzhalter * und ? verwendet werden. Sie bewirken, daß die entsprechenden Felder vom Eingabedateinamen übernommen werden.
`/S`	Bewirkt, daß auch Dateien aus Unterverzeichnissen kopiert werden. Das Zielverzeichnis wird, falls notwendig, angelegt.
`/L`	Bewirkt, daß die Dateinamen nach dem Kopieren ausgedruckt werden.

Beschreibung

Die Dateien, die mit *`old file spec`* angegeben werden, werden nach *`new file spec`* kopiert. Falls *`new file spec`* weggelassen wird, werden die Dateien in das aktuelle Verzeichnis kopiert.

`COPY` unterstützt die Verwendung von Gerätenamen anstelle von Dateinamen. Somit kann eine Datei auf ein Gerät, die Eingaben von einem Gerät auf eine Datei oder Eingaben von einem Gerät auf ein anderes Gerät kopiert werden. Die möglichen Gerätenamen sind im Betriebsmittelkatalog eingetragen. Zusätzlich kann der Name CON benutzt werden, um die gegenwärtigen Konsole anzugeben.

Beispiele

```
COPY A:\REPORT.LIS
```

```
Kopiert die Datei A:\REPORT.LIS in das aktuelle Verzeichnis.
```

```
copy A:\*.* /S
```

Kopiert alle Dateien von Laufwerk A: in das aktuelle Verzeichnis. Unterverzeichnisse werden mitkopiert.

```
COPY TEST.* OLD_TEST.*
```

Kopiert alle Dateien im aktuellen Verzeichnis mit dem Namen TEST und beliebiger Erweiterung in die Dateien mit dem Namen OLD_TEST und die jeweils gleiche Erweiterung. Z.B. TEST.C wird nach OLD_TEST.C, TEST.EXE wird nach OLD_TEST.EXE kopiert.

```
COPY CLISTART.BAT CON
```

Kopiert die Datei CLISTART.BAT zur Konsole.

```
COPY CLISTART.BAT BYT_LPT1
```

Kopiert die Datei CLISTART.BAT zum Druckeranschluß LPT1, vorausgesetzt, daß BYT_LPT1 im Betriebsmittelkatalog als Type `rio byte` eingetragen ist (Abfrage mit dem Low-Level Debugger-Befehl: `DIR unit long`.

```
COPY CON TEMP.BAT
```

Kopiert alle weiteren Eingaben an der Konsole in die Datei TEMP.BAT, bis die Eingabe <Ctrl>+<Z> erkannt wird.

```
COPY CON BYT_LPT1
```

Kopiert alle weiteren Eingaben an der Konsole zum Druckeranschluß LPT1, bis die Eingabe <Ctrl>+<Z> erkannt wird, vorausgesetzt BYT_LPT1 ist im Betriebsmittelkatalog als Type `rio byte` eingetragen.

Die Zeichen werden zeilenweise kopiert.

B.6 CPRI

Funktion Task-Priorität eines CLI-Jobs ändern

Inline-Kommando

Syntax cpri <Job number> <Priority>

Parametername	Bedeutung
`Job number`	Job-Nummer, die beim Starten vergeben wurde. Die Nummer wird mit dem Kommando SYSTAT angezeigt.
`Priority`	Neue Task-Priorität.

Beschreibung

Die Priorität des mit *`Job number`* spezifizierten Hintergrund-Jobs wird gleich *`Priority`* gesetzt.

Falls ein Batchdatei als Hintergrund-Job ausgeführt wird, ist es möglich, daß der ursprünglich gestartete Job auf die Beendigung eines weiteren CLI-Jobs wartet.

In diesem Falle wird lediglich die Priorität des ursprünglich gestarteten Hintergrund-Jobs geändert. Es ist jedoch möglich, die Priorität des untergeordneten Jobs, durch die Angabe seiner eigenen Job-Nummer zu ändern. Die Job-Nummer des untergeordneten Jobs wird ebenfalls vom Kommando SYSTAT angezeigt.

Beispiel

```
CPRI 3 64
```

Dieses Beispiel setzt die Priorität des Hintergrund-Jobs Nummer 3 auf 64.

B.7 DATE

Funktion Aktuelles Datum anzeigen oder ändern

nachladbares Kommando

Syntax date [<date>]

Parametername	Bedeutung
date	Neues Datum im Format <Tag>-<Monat>-<Jahr>.

Beschreibung

Falls DATE ohne Parameter eingegeben wird, wird das aktuelle Datum angezeigt und darauf gewartet, daß der Benutzer ein neues Datum eingibt. Durch einfaches Drücken der Eingabe-Taste bleibt das Datum unverändert.

Das Datum darf nur auf Werte zwischen dem 01-01-1980 und 31-12-2059 gesetzt werden.

Der Monat kann entweder als eine Zahl oder eine drei-Buchstaben-Abkürzung (in Englisch) eingegeben werden.

Das Jahr kann entweder als eine zweistellige oder eine vierstellige Zahl eingegeben werden. Zweistellige Zahlen im Bereich 80 bis 99 entsprechen den Jahren 1980 bis 1999. Zweistellige Zahlen im Bereich 00 bis 59 entsprechen den Jahren 2000 bis 2059.

Sie können das Jahr, oder Monat und Jahr weglassen. Die Defaultwerte werden vom aktuellen Datum genommen.

Beispiele

```
DATE
Current date is Monday 17-FEB-1992
Enter new date (dd-mm-yy):

DATE 18-FEB-92

DATE 18-FEB

DATE 18-2-1992

DATE 19
```

B.8 DEL

Funktion Eine oder mehrere Dateien löschen

nachladbares Kommando

Syntax del <file spec> [/P] [/L]

Parametername	Bedeutung
file spec	Die Datei oder Dateien, die zu löschen sind. Beim Dateinamen und der Erweiterung können die Platzhalter * und ? verwendet werden.
/P	Bewirkt, daß bei jeder Datei nachgefragt wird, ob sie wirklich gelöscht werden soll.
/L	Bewirkt, daß die Dateinamen nach dem Löschen ausgedruckt werden.

Beschreibung

Dieses Kommando löscht die angegebene(n) Datei(en).

Beispiel

```
DEL TEMP.*
```

Alle Dateien im aktuellen Verzeichnis mit dem Namen TEMP werden gelöscht.

```
DEL *.TMP
```

Alle Dateien im aktuellen Verzeichnis mit der Erweiterung TMP werden gelöscht.

```
DEL *.TMP /P
```

Für jede Datei im aktuellen Verzeichnis mit der Erweiterung TMP wird gefragt, ob sie gelöscht werden soll. Falls eine Bestätigung erfolgt, wird sie tatsächlich gelöscht.

```
DEL A:\WORK\REPORT.LIS
```

Die Datei A:\WORK\REPORT.LIS wird gelöscht.

B.9 DEVICE

Funktion **Treiber laden oder Treiber-Unit erzeugen**

nachladbares Kommando

Syntax **Treiber laden:**

DEVICE <pfad name> [<unit name> [<unit parameter>]]

Treiber-Unit erzeugen:

DEVICE <katalog name> <unit name> [<unit parameter>]

Parametername	Bedeutung
pfad name	Vollständiger Pfadname der Treiber–Datei, z.B. \M7RMOS32\3964.DRV
katalog name	Name, unter welchem der Treiber in den Katalog eingetragen wurde: 3964 bzw. SER8250. Groß–/Kleinschreibung beachten.
unit name	Benutzerdefinierter Name der Unit, die erzeugt werden soll.
unit prameter	Treiberspezifische Parameter

Treiberspezifische Parameter

Die treiberspezifischen Parameter werden beim Erzeugen der Unit benötigt. Sie bestehen aus einer Folge von Zeichenketten (Strings), die durch Leerzeichen getrennt sind. Die generische Form einer solchen Zeichenkette lautet:

```
<Schluesselwort>:<par_1>-<par_2>-...<par_n>...
```

Eine Zeichenkette besteht aus einem Schlüsselwort gefolgt von Unterparametern, die durch Bindestriche getrennt werden.

Parameter für 3964 und SER8250

Die treiberspezifischen Parameter für 3964 lauten:

```
IRQ:<irq nummer> BASE:<i/o basis adresse> [MODE:<baudrate>-<par-
ity>-<data bit>-<stop bit>] [PROT:<protokoll>-<master/slave>]
```

Die treiberspezifischen Parameter für SER8250 lauten:

```
IRQ:<irq nummer> BASE:<i/o basis adresse> [MODE:<baudrate>-<par-
ity>-<data bit>-<stop bit>] [BUFFER:<groesse>]
```

Parametername	Bedeutung
IRQ:<irq nummer>	Interrupt-Nummer der Schnittstelle, über die der Treiber kommunizieren soll, z.B.: IRQ:4 für COM1.
BASE:<i/o basis adresse>	I/O Basis-Adresse der Schnittstelle, über die der Treiber kommunizieren soll, z.B.: BASE:0x3F8 für COM1.
MODE:<baud>–<parity>–<data>–<stop>	Kommunikations-Parameter. Fehlt die Angabe von MODE, dann wird die Voreinstellung MODE:19200–N–8–1 genommen
baud	Übertragungsgeschwindigkeit (z.B. 19200 bit/s)
parity	Parity-Bit (z.B. N für 'none', E für 'even', O für 'odd', M für 'mark', S für 'space')
data	Anzahl der Daten-Bits
stop	Anzahl der Stop-Bits
PROT:<protokoll>–<master>	Protokoll-Parameter, nur für 3964-Treiber. Fehlt die Angabe von PROT, dann wird die Voreinstellung PROT:1–1 genommen
protokoll	Protokoll-Auswahl: 1 für 3964R, 0 für 3964
master	Auswahl Master oder Slave: 1 für Master, 0 für Slave
BUFFER:<groesse>	Größe des Hintergrundpuffers in Byte, nur für SER8250-Treiber. Voreinstellung: BUFFER:256.

Beschreibung

Das DEVICE-Kommando muß in folgender Reihenfolge aufgerufen werden – in der Datei RMOS.INI oder im CLI:

1. Laden des Treibers anhand des Pfadnamens der Treiberdatei.
   DEVICE <pfad name> [<unit name> [<unit parameter>]]
   Beim Laden wird der Treiber in den Betriebsmittelkatalog eingetragen. Optional kann beim Laden des Treibers auch eine Treiber-Unit erzeugt werden. Ein Treiber kann nur einmal geladen werden.
2. Erzeugen einer Unit anhand des Katalognamens des Treibers.
   DEVICE <katalog name> <unit name> [<unit parameter>]
   Mit weiteren DEVICE-Aufrufen für diesen Treiber können nur noch Units erzeugt werden. Dafür müssen der Katalogname des Treibers und die Unit–Parameter angegeben werden.

   Es kann nur eine Unit pro Hardware-Schnittstelle existieren. Ein DEVICE-Aufruf, der eine Unit für eine bereits bestehende Schnittstelle (IRQ und BASE) unter einem neuen Namen erzeugt, überschreibt damit die alte Unit.

Hinweis

Die Hardware-Angaben zu den seriellen Schnittstellen für M7 finden Sie in den Kapiteln "Zentralbaugruppe CPU 388-4" und "Schnittstellenmodul IF 962-COM" im Handbuch "M7-300, Aufbauen CPU-Daten" oder im Kapitel "Schnittstellenmodul IF 962-COM" im Handbuch "S7-400, M7-400, Baugruppendaten".

Beispiel

3964-Treiber laden ohne eine Unit zu erzeugen:

```
DEVICE \M7RMOS32\3964.DRV
```

Treiber-Unit für 3964 an COM1 erzeugen:

```
DEVICE 3964 3964_COM1 IRQ:4 BASE:0x3F8 MODE:19200–N–8–1 PROT:1–1
```

SER8250-Treiber laden:

```
DEVICE \M7RMOS32\SER8250.DRV
```

Treiber-Unit für SER8250 an COM2 erzeugen:

```
DEVICE SER8250 COM2 IRQ:3 BASE:0x2F8 MODE:9600–N–8–1
```

B.10 DIR

Funktion Inhaltsverzeichnis auflisten

nachladbares Kommando

Syntax dir [<Dateiname>] [/P] [/W] [/S]

Parametername	Bedeutung
Dateiname	Das Verzeichnis oder die Dateien, die aufgelistet werden sollen. Beim Dateinamen können die Platzhalter * und ? verwendet werden.
/P	Bewirkt, daß das Verzeichnis seitenweise ausgegeben wird.
/W	Bewirkt, daß das Verzeichnis im Breitformat, in fünf Spalten, ausgegeben wird.
/S	Bewirkt, daß Unterverzeichnisse mit ausgegeben werden.

Beschreibung

Mit DIR werden die Dateien in einem Verzeichnis oder einem Verzeichnisbaum aufgelistet. Im Normalfall, werden die Dateinamen mit Datum, Uhrzeit und Größe aufgelistet. Mit dem Schalter /W werden nur die Dateinamen aufgelistet. Am Ende der Liste wird die Gesamtgröße aller aufgelisteten Dateien und der freie Platz auf dem Laufwerk ausgegeben. Falls kein Dateiname angegeben wird, wird das aktuelle Verzeichnis aufgelistet. Dateiname kann ein beliebigen (gültiger) Pfad mit oder ohne Laufwerksangabe sein.

Beispiel

```
DIR
DIR *.TXT
DIR TEMP.*
DIR A:\*.*
DIR /W /S
DIR ..
```

B.11 DISMOUNT

Funktion Datenträger abmelden

Inline-Kommando

Syntax dismount <Laufwerk>: [/F]

Parametername	Bedeutung
Laufwerk	Das Laufwerk, in dem sich der Datenträger befindet, der abgemeldet werden soll.
/F	Das angegebene Laufwerk wird zwangsabgemeldet.

Beschreibung

Mit dem RMOS Dateiverwaltungssystem (HSFS) ist es grundsätzlich notwendig, einen Datenträger abzumelden, bevor Sie ihn aus dem Laufwerk nehmen oder das System abschalten.

Ein Datenträger wird mit DISMOUNT beim HSFS abgemeldet, dies ist jedoch nur möglich, wenn keine Dateien auf dem Laufwerk offen sind. Mit dem Schalter /F wird das Laufwerk auch abgemeldet, wenn Dateien auf dem Laufwerk offen sind. Diese Dateien werden ordnungsgemäß geschlossen. Programme, die diese Kanalnummern noch verwenden, erhalten den Fehler H_NO_SUCH_CHANNEL zurückgemeldet.

Grundsätzlich wird der Datenträger abgemeldet und nicht das Laufwerk, obwohl das Laufwerk als Parameter angegeben wird.

Befindet sich das Laufwerk (oder ein Verzeichnis des Laufwerks) im Suchpfad, kann das Volume bei Eingabe eines Kommandos wieder angemeldet sein.

Beispiel

```
DISMOUNT A:
```

B.12 ECHO

Funktion **Anzeigen der Befehlszeile bei Batchdatei-Bearbeitung ein- oder ausschalten, oder eine Meldung ausgeben**

Inline-Kommando

Syntax **echo [ON | OFF | <Meldung>]**

Beschreibung

Die Voreinstellung des CLI ist ECHO OFF.

Bei der Bearbeitung von Batchdateien wird die Einstellung vom CLI übernommen, es werden daher die Kommandos einer Batchdatei nicht angezeigt. Das Anzeigen der Befehle kann mit `ECHO ON` eingeschaltet werden, mit `ECHO OFF` wieder ausgeschaltet werden.

`ECHO <Meldung>` schreibt `<Meldung>` nach `stdout`. Durch die Umlenkung von `stdout` auf eine Datei können Textzeilen in eine Datei geschrieben werden.

Beispiele

```
ECHO OFF
ECHO ON
ECHO Start of Batchfile DO_IT.BAT
ECHO Processing completed. >> work.log
```

B.13 ERROR

Funktion Errorlevel und seine Bedeutung anzeigen

nachladbares Kommando

Syntax error [<Errorlevel>]

Parametername	Bedeutung
`Errorlevel`	Statuswert als vierstellige Hexadezimal-Zahl.

Beschreibung

Nach dem Ausführen eines CLI-Kommandos oder eines beliebigen Programmes wird der Errorlevel gleich dem Exitstatus des Kommandos bzw. Programms.

ERR0R ohne Parameter liest den gegenwärtigen Errorlevel und druckt ihn als eine Hex-Zahl aus. Falls möglich wird außerdem die Bedeutung des Errorlevels ausgedruckt.

ERR0R `<Errorlevel>` druckt die Bedeutung des als Parameter angegebenen Errorlevels aus.

In beiden Fällen wird ERR0R mit dem Exitstatus abgeschlossen, der gleich dem bisherigen Errorlevel ist, d.h., der Errorlevel wird von diesem Kommando nicht verändert.

ERR0R geht von einem 16-Bit-Errorlevel mit folgendem Format aus:

- 0 = Success

sonst

- High-Byte (Bits 8-15)
  Hex 81 : Tasklader-Errorcode
  Hex 82 : CLI-Errorcode
  Hex 83 : HSFS-Errorcode
- Low-Byte (Bits 0-7)
  Errorcode

Hinweis

Der High-Byte-Wert Hex 83 für HSFS-Errorcodes wird vom CLI erzeugt.

Beispiel

```
ERR0R
```

B.14 EXIT

Funktion CLI-Task beenden

Inline-Kommando

Syntax exit

Beschreibung

Die CLI-Task wird durch `EXIT` beendet.

Bemerkungen

`EXIT` endet mit dem Exitstatus E_CLI_EXIT_STATUS, worauf die CLI-Task beendet wird. Durch die Verwendung dieses Exitstatus in Ihren Programmen können Sie die CLI-Task beenden.

Beispiel

```
EXIT
```

B.15 FORMAT

Funktion Diskette oder Festplatte formatierten

nachladbares Kommando

Syntax format <Laufwerk>: /F

Parametername	Bedeutung
Laufwerk	Das Laufwerk, in dem sich der Datenträger befindet, der formatiert werden soll.

Beschreibung

Formatiert eine Diskette oder Festplatte und legt das Hauptverzeichnis an. Ein Laufwerk kann nur formatiert werden, wenn es **nicht** angemeldet ist.

Es können nur 1,44-Mbyte-Disketten formatiert werden.

Die Option /F erlaubt das Formatieren des Datenträgers, auch wenn im HSFS das Bit für Formatierschutz gesetzt ist (siehe CRUN-Aufrufe **changevib** und **createvib**).

Beispiel

```
FORMAT A:
```

B.16 FTLFORM

Funktion Memory Card oder OSD formatieren

nachladbares Kommando

Syntax ftlform <*drive:*> [*/n*] [/s [<file>]] |/b] [/m] [/c[<file>]] [/?|/h]

Parametername	Bedeutung
drive:	Laufwerk mit der zu formatierenden Memory Card. (M0 bzw. M1)
n	Während des Formatiervorgangs werden keine Meldungen ausgegeben.
s[<file>]	Die mit *file* angegebene Systemdatei wird nach dem Formatieren auf die Memory Card übertragen.
b	Das Kommando läuft im Batch Modus ab
m	Der Master Boot Record wird kopiert
c[<file>]	Die mit *file* angegebene Konfigurationsdatei wird geladen.
? bzw. *h*	Ausgabe der Kommando-Syntax

Beschreibung

Das Kommando FTLFORM formatiert die Memory Card in dem mit *drive* angegebenen Laufwerk (M0 bzw. M1). Mit dem Parameter *?* bzw. *h* wird eine Hilfe zur Kommandosyntax ausgegeben.

Mit dem Parameter *n* werden die Meldungen des Kommandos während des Formatiervorgangs nicht ausgegeben.

Mit dem Parameter *s* überträgt FTLFORM nach Abschluß des Formatiervorgangs die mit *file* angegebene Systemdatei auf die Memory Card. Wird keine Datei angegeben, wird die Datei RM3_PC1.SYS im aktuellen Verzeichnis auf die Memory Card übertragen. Zusätzlich wird auf der Memory Card ein Bootsektor entsprechend *drive:* eingerichtet. Damit kann die Memory Card als Boot-Medium eingesetzt werden.

Mit dem Parameter *b* läuft das Kommando im Batch Modus ab.

Mit dem Parameter *m* wird der Master Boot Record kopiert.

Mit dem Parameter *c* wird die mit *file* angegebene Konfigurationsdatei geladen. In der Default-Einstellung ist dies die Datei FTLFORM.CFG. Die Konfigurationsdatei legt die weiteren Bedingungen für die anschließende Formatierung fest.

In der Konfigurationsdatei gibt es drei Abschnitte.

[SYS_FILES] In diesem Abschnitt werden die Systemdateien mit vollständiger Pfadangabe eingetragen. Maximal sind

drei Dateien erlaubt.

[BOOT_RECORD] In diesem Abschnitt wird der Name des (binären) Boot-Laders (512 Byte) mit vollständigem Pfad angegeben. Der Boot-Lader wird nach dem Formatiervorgang in den Boot-Sektor der Memory Card übertragen.

[MASTER_BOOT_RECORD]
In diesem Abschnitt wird der Name des (binären) Master Boot-Laders mit vollständigem Pfad angegeben.

Die Angaben in den Abschnitten [SYS_FILES] und [BOOT_RECORD] müssen vorgenommen werden, da auf keine Default-Einstellungen zurückgegriffen wird.

Ebenso dürfen keine leeren Abschnitte angegeben werden, d.h. wird der Eintrag für [BOOT_RECORD] gestrichen, ist auch die Abschnittsbezeichnung selbst zu löschen.

Kommentare für zusätzliche Informationen oder Hinweise werden durch ein "#" gekennzeichnet und dadurch bei der Abarbeitung nicht berücksichtigt.

Beispiel für eine Konfigurationsdatei:

```
[SYS_FILES]
c:\rmos3\boot\rmos.sys

[BOOT_RECORD]
c:\rmos3\boot\rmosload

#Die Dateien enthalten den Software-Stand V3.12.09
```

Beispiel

```
ftlform -s M0:
```

Vor dem Formatiervorgang erfolgt eine Sicherheitabfrage.

```
Application Identification of drive M0: is M7-DOS
Are you sure to format drive M0: ? [y/N]
```

Default-Vorgabe ist N(ein).
Geben Sie y an und bestätigen Sie mit Return. Danach wird die Memory Card formatiert und abschließend eine Meldung über das Ergebnis des Formatiervorgangs ausgegeben.

```
Formatting M0: ...
M0: formatted: 829 KB useable
```

Danach wird die Default-Systemdatei übertragen und die Kennung der Memory Card auf M7-DOS gesetzt.

```
Copying system files...
     RM3_PC1.SYS
Setting application ID of drive M0: to M7-DOS
```

B.17 HELP

Funktion Kommandoübersicht oder Syntaxerklärung für einzelne Kommandos ausgeben

nachladbares Kommando

Syntax help [<Kommando>]

Parametername	Bedeutung
Kommando	CLI-Kommando, über das Information gewünscht wird.

Beschreibung

HELP ohne Parameter gibt eine Übersicht aller CLI-Kommandos aus.

HELP <Kommando> gibt eine Beschreibung des angegebenen Kommandos aus.

Beispiele

```
HELP
HELP DIR
```

B.18 MD

Funktion Neues Verzeichnis anlegen

nachladbares Kommando

Syntax md <Verzeichnis>

Parametername	Bedeutung
Verzeichnis	Verzeichnis, das angelegt werden soll in der Standard-Schreibweise, mit oder ohne Laufwerksangabe.

Beschreibung

Mit MD wird ein neues Verzeichnis angelegt. Das Verzeichnis kann relativ zum Hauptverzeichnis oder zum aktuellen Verzeichnis spezifiziert werden. Das übergeordnete (Parent) Verzeichnis kann mit ”..” angegeben werden.

Beispiele

```
MD TEMP
```

Legt ein Verzeichnis TEMP als Unterverzeichnis des aktuellen Verzeichnisses an.

```
MD \TEMP
```

Legt ein Verzeichnis TEMP als Unterverzeichnis des Hauptverzeichnisses im aktuellen Laufwerks an.

```
MD ..\TEMP
```

Legt ein Verzeichnis TEMP als Unterverzeichnis des Parent-Verzeichnisses des aktuellen Verzeichnisses an.

```
MD A:\TEMP
```

Legt ein Verzeichnis TEMP als Unterverzeichnis des Hauptverzeichnis des Laufwerks A: an.

B.19 MOUNT

Funktion Datenträger anmelden oder Zustand aller Laufwerke anzeigen

Inline-Kommando

Syntax mount [<Laufwerk>:]

Parametername	Bedeutung
Laufwerk	Das Laufwerk, in dem sich der Datenträger befindet, der angemeldet werden soll.

Beschreibung

Mit dem RMOS Dateiverwaltungssystem (HSFS) ist es grundsätzlich notwendig, einen Datenträger vor dem Verwenden anzumelden (MOUNT) und auch nachher abzumelden (DISMOUNT). Mit dem Befehl MOUNT <Laufwerk>: wird versucht, einen Datenträger im angegebenen Laufwerk anzumelden.

Mit MOUNT ohne Parameter wird eine Liste aller konfigurierten Laufwerke mit ihrem gegenwärtigen Zustand (MOUNTED bzw. DISMOUNTED) ausgegeben.

Grundsätzlich wird der Datenträger angemeldet und nicht das Laufwerk, obwohl das Laufwerk als Parameter angegeben wird.

Beim Wechseln (cd) auf ein anderes Laufwerk wird dieses automatisch angemeldet.

Beispiel

```
MOUNT A:
```

B.20 NPX – nicht für weitere Entwicklungen

Funktion **Präfix zum Starten eines Programms, das den numerischen Coprozessor verwendet**

Hinweis

Dieses Kommando ist ab Version V2.0 nicht mehr notwendig, da automatisch erkannt wird, wenn der numerische Coprozessor für eine Task verwendet werden muß. Das Kommando wird weiterhin aus Kompatibilitätsgründen unterstützt, soll aber **nicht für weitere Entwicklungen** verwendet werden.

B.21 PATH

Funktion Suchpfad für nachladbare CLI-Kommandos festlegen

Inline-Kommando

Syntax path [<Verzeichnis>] [;<Verzeichnis>...]

path ;

Parametername	Bedeutung
Verzeichnis	Gültiges Verzeichnis in der Standard-Schreibweise oder %PATH%

Beschreibung

Mit PATH wird ein Suchpfad für nachladbare CLI Kommandos definiert. Der Suchpfad besteht aus eine Liste von Verzeichnissen, die jeweils durch ein Semikolon getrennt sind.

Wenn Sie einen Befehl eingeben, sucht der CLI wie folgt nach einem Inline- oder nachladbaren Kommando:

- Zuerst wird nach einem Inline-Kommando gesucht.
- Danach wird nach einer ausführbaren Datei oder einer Batchdatei im aktuellen Verzeichnis gesucht.
- Danach wird der Reihe nach durch alle Verzeichnisse im Suchpfad nach einer ausführbaren Datei oder einer Batchdatei gesucht.

Die Suche wird abgebrochen, sobald das gesuchte Inline- oder nachladbare Kommando gefunden wird.

PATH ohne Parameter zeigt den derzeitigen Suchpfad an.

PATH ; definiert einen leeren Suchpfad.

Die Angabe %PATH% entspricht dem derzeitigen Suchpfad. Damit kann der Suchpfad erweitert werden (siehe Beispiele).

Hinweis

Ein Semikolon am Ende des Suchpfads ist erlaubt aber nicht notwendig. Alle Verzeichnisse im Suchpfad sollten vollständige Pfad-Spezifikationen einschließlich Laufwerksangabe sein.

Der Suchpfad und PATH funktionieren grundsätzlich wie bei DOS.

Beispiel

```
PATH C:\CLI;C:\TOOLS
```

Ein Suchpfad wird definiert, der aus den zwei Verzeichnissen C:\CLI und C:\TOOLS besteht.

```
PATH %PATH%;C:\PROJECT
```

Der Suchpfad wird um das Verzeichnis C:\PROJECT erweitert. Falls beispielsweise der bisherige Suchpfad C:\CLI;C:\TOOLS war, wird der neue Suchpfad auf C:\CLI;C:\TOOLS;C:\PROJECT erweitert.

```
PATH ;
```

Definiert einen leeren Suchpfad.

```
PATH
```

Zeigt den derzeitigen Suchpfad an.

B.22 PING

Funktion **Prüfen, ob ein TCP/IP Netzwerkteilnehmer aktiv ist**

nachladbares Kommando

Syntax **ping <host>**
ping [-s] <host> [<data_size> [<count>]]

Parametername	Bedeutung
host	Name oder IP-Adresse des Kommunikationsteilnehmers
–s	Bewirkt, daß mehrere Datagramme gesendet werden
data_size	Legt die Größe eines Datagramm–Pakets in Bytes fest; die voreingestellte Paketgröße beträgt 64 Bytes.
count	Optional kann mit <count> festgelegt werden, wieviele Anforderungen gesendet werden sollen.

Beschreibung

PING sendet ein Datagramm, um eine Antwort vom spezifizierten <host> hervorzurufen. Wenn <host > antwortet, gibt PING die Meldung

```
<host> is alive
```

auf der aktuellen Konsole (standard output) aus und beendet sich. Ansonsten erscheint nach 5 Sekunden die Meldung

```
no answer from <host>
```

Wenn der Schalter **-s** angegeben ist, sendet PING ein Datagramm pro Sekunde, und schreibt jeweils eine Ausgabezeile für jede erhaltene Antwort. Für fehlende Antworten werden keine Meldungen ausgegeben. In dieser zweiten Form berechnet PING die Übertragungszeiten und die Statistiken der Datenverluste und zeigt diese Informtionen als Übersicht an.

Wenn PING zur Eingrenzung von Fehlern genutzt wird, sollte zuerst der lokale Host geprüft werden, um festzustellen, ob die lokale Netzwerkschnittstelle aktiv ist.

Beispiel

```
ping 142.120.12.320
```

B.23 PROMPT

Funktion Eingabeaufforderung des CLI ändern

Inline-Kommando

Syntax prompt [[<Text>][$<Zeichen>]...]

Parametername	Bedeutung
Text	Text der Eingabeaufforderung
$<Zeichen>	Steuerzeichen, siehe unten

Beschreibung

Die CLI-Eingabeaufforderung (Prompt) wird geändert. Der Prompt wird im Environment gespeichert.

Der Prompt kann Steuerzeichen im Format "$<Zeichen>" beinhalten, die vom CLI expandiert werden. Die folgenden Steuerzeichen sind möglich :

$q	Das = Zeichen
$$	Das $ Zeichen
$t	Die aktuelle Uhrzeit
$d	Das aktuelle Datum
$p	Das aktuelle Verzeichnis einschließlich Laufwerk
$v	Die aktuelle CLI Version
$g	Das > Zeichen
$l	Das < Zeichen
$b	Das | Zeichen
$_	Eine Zeilenvorschub
$e	Escape
$h	Backspace

Wenn Sie PROMPT ohne Parameter eingeben, wird der Environment-String PROMPT gelöscht. Danach wird die Default-Eingabeaufforderung '>' verwendet.

Beispiel

```
PROMPT $P$G
```

setzt die Eingabeaufforderung gleich <aktuelles Verzeichnis>>, beispielsweise:

```
C:\MY_DIR>
```

B.24 RD

Funktion Leeres Unterverzeichnis löschen

nachladbares Kommando

Syntax rd <Verzeichnis>

Parametername	Bedeutung
`Verzeichnis`	Verzeichnis, das entfernt werden soll, in Standard-Schreibweise mit oder ohne Laufwerksangabe.

Beschreibung

Falls

- das angegebene Verzeichnis leer ist und
- das Verzeichnis von keiner Task als offene Datei gehalten wird,

wird das Verzeichnis gelöscht.

Beispiel

```
RD TEMP
```

Das Unterverzeichnis TEMP wird aus dem aktuellen Verzeichnis gelöscht.

B.25 RDISK

Funktion Bootssektor auf Festplatte einrichten

nachladbares Kommando

Syntax RDISK [RM3_PC1.SYS]

Parametername	Bedeutung
RM3_PC1.SYS	Name des Boot-Files. Die Angabe ist optional; der Name **RM3_PC1.SYS** ist jedoch fest.

Beschreibung

RDISK richtet auf der C:-Partition der Festplatte einen Bootsektor ein. Als Bootfile wird die Systemdatei RM3_PC1.SYS verwendet.

Bevor Sie das Kommando aufrufen, müssen Sie die Festplatte mit `HDPART` partitionieren und mit `FORMAT` formatieren.

Vor der Ausführung des Kommandos wird eine Bestätigung angefordert, z.B. bei der ersten Installation des Bootsektors

```
Install boot sector on drive C: (y/n)?
```

oder wenn bereits ein Bootsektor installiert ist:

```
Overwrite active boot sector on drive C: (y/n)?
```

In beiden Fällen wird das Kommando nur nach Bestätigung mit `y` ausgeführt.

`RDISK` beendet sich mit der Meldung:

```
RMOS PC1 bootsector installed for file RM3_PC1.SYS boot
```

bei erfolgreicher Installation des Bootsektors oder

```
RMOS PC1 bootsector not installed
```

im Fehlerfall oder wenn `RDISK` vom Benutzer vorzeitig abgebrochen wurde.

B.26 RENAME

Funktion **Eine oder mehrere Dateien bzw. Verzeichnisse umbenennen**

nachladbares Kommando

Syntax **rename <old file spec> <new file spec> [/L]**

Parametername	Bedeutung
`old file spec`	Datei bzw. Verzeichnis oder Dateien bzw. Verzeichnisse, die umzubenennen sind. Beim Namen und der Erweiterung können die Platzhalter * und ? verwendet werden.
`new file spec`	Der neue Dateiname bzw. Verzeichnisname. Beim Dateinamen bzw. Verzeichnisnamen und der Erweiterung können die Platzhalter * und ? verwendet werden. Sie bewirken, daß die entsprechenden Felder vom Eingabedateinamen übernommen werden. Es darf weder ein Verzeichnis noch ein Laufwerk beim neuen Dateinamen bzw. Verzeichnisnamen angegeben werden.
`L`	Bewirkt, daß die Dateinamen bzw. Verzeichnisnamen nach dem Umbenennen ausgegeben werden.

Beschreibung

Die Datei bzw. Verzeichnis oder Dateien bzw. Verzeichnisse, die mit `<old file spec>` angegeben werden, werden nach `<new file spec>` umbenannt. Die Dateien bzw. Verzeichnisse bleiben im gleichen Verzeichnis.

Beispiel

```
RENAME TEST.PLM TEST.C
```

Die Datei TEST.PLM wird nach TEST.C umbenannt.

```
RENAME DUMMY.DIR TEST.DIR
```

Das Verzeichnis DUMMY.DIR wird nach TEST.DIR umbenannt.

```
RENAME TEMP.* SAVE.*
```

Alle Dateien und Verzeichnisse im aktuellen Verzeichnis mit dem Namen TEMP werden in SAVE umbenannt. Die Erweiterungen werden nicht geändert.

B.27 SCANDISK

Funktion Datenträger überprüfen und korrigieren

nachladbares Kommando

Syntax scandisk <Laufwerk>: [/F | /A] [/T]

Parametername	Bedeutung
Laufwerk	Das Laufwerk, in dem sich der Datenträger befindet, der überprüft werden soll.
/F	Fehler anzeigen und nach Rückfrage korrigieren
/A	Fehler anzeigen und automatisch korrigieren
/T	Keine Anzeige der aktuell untersuchten Dateien

Beschreibung

scandisk überprüft das ausgewählte Medium - Festplatte, Diskette oder Memory Card - auf logische Konsistenz und führt interaktiv die Korrektur durch.

scandisk ist ein Kommando für reines M7 RMOS32. Bitte beachten Sie, daß in der Systemvariante M7 RMOS32 mit MS-DOS die Laufwerke in der Regel unter DOS verwaltet werden und unter DOS geprüft werden müssen. Generell muß die Konsistenz von Netzlaufwerken am jeweiligen Hostsystem sichergestellt werden.

Für die Dauer des Programmlaufs meldet **scandisk** das ausgewählte Laufwerk ab und reserviert sich den Zugriff darauf. Andere Programme können in dieser Zeit das Laufwerk nicht nutzen.

scandisk zeigt den Testfortschritt an und meldet gefundene Fehler. Sofern nicht die Option **/T** eingestellt ist, werden auch die Namen aller gerade untersuchten Dateien angezeigt. Eine kurze Statistik über den Laufwerksinhalt folgt am Ende.

Wenn Fehler gefunden werden, gibt es folgende Möglichkeiten:

Option	Fehlerbehandlung
keine	Der Aufruf zeigt die gefundenen Fehler an, ohne Korrekturen vorzunehmen
/F	Jeder gefundene Fehler wird angezeigt und es wird abgefragt, ob die vorgeschlagene Korrektur durchgeführt werden soll oder nicht
/A	Jeder gefundene Fehler wird angezeigt und automatisch korrigiert

Fehlerarten und Korrekturen

scandisk kann folgende Fehlerarten finden und Korrekturen vorschlagen, bzw. durchführen:

Fehlerart	Korrekturvorschlag [/F] bzw. Korrektur [/A]
Der Bootsektor enthält ungültige Werte über Größe, FAT-Länge oder Root-Verzeichnis.	Abbruch: Das Laufwerk enthält wahrscheinlich kein Dateisystem im DOS-Format.
Abweichung der FAT-Kopien von der ersten FAT.	Erneuerung der Sicherungskopien durch die erste FAT.
Ein Dateiname enthält ein ungültiges Zeichen: [\ | / | . | : | ? | ∗] oder Steuerzeichen.	Ersetzen der ungültigen Zeichen, soweit möglich, durch eines der folgenden : [@ | $ | %]. Ist dies nicht möglich, wird der Dateiname in eine Checkdatei *) gewandelt.
Crosslinks (mehrfach benutzte Cluster).	Die Clusterkette ab dem mehrfach benutzten Cluster wird kopiert und in die Datei eingebaut.
Eine Clusterkette enthält Einträge, die über die Größe des Mediums hinausreichen.	Das Dateiende wird hinter das letzte gültige Cluster gesetzt und die Dateilänge angepaßt.
Eine Datei hat die Länge 0, besitzt aber ein gültiges Startcluster.	Die Dateilänge wird auf die Größe gesetzt, die sich aus der Clusterverkettung ergibt.
Eine Datei hat eine Länge ungleich 0, besitzt aber kein gültiges Startcluster.	Die Dateilänge wird zu 0 gesetzt.
Die Dateilänge ist größer als die Datenmenge der eingeketteten Cluster.	Die Dateigröße wird angepaßt auf die Größe der eingeketteten Cluster.
Die Dateilänge umfaßt nicht die Datenmenge der Clusterverkettung.	Die Dateigröße wird angepaßt. Sie wird vergrößert und beinhaltet den kompletten letzten Cluster.
Die Clusterverkettung einer Datei zeigt auf eigene Cluster (rekursive Cluster).	Das Dateiende wird hinter das letzte gültige Cluster gesetzt und die Dateilänge angepaßt.
Die Clusterverkettung zeigt auf einen unbenutzten Cluster.	Der neue Cluster wird in die Kette integriert und die Dateilänge wird entsprechend angepaßt.
Das Verzeichnisattribut ist gesetzt, die Dateilänge ist jedoch nicht passend und die ersten Einträge sind nicht “.” und “..”.	Konvertierung in eine Datei.
Verlorene Dateien: Es sind Cluster belegt, die keiner Datei zugeordnet sind.	Die jeweils zusammenhängenden Cluster werden als Checkdateien *) im Root-Verzeichnis \ abgelegt.
*) eine Checkdatei hat folgende Namensstruktur: “FILExxxx.CHK” wobei xxxx ein numerischer Wert ist, der fortlaufend hochgezählt wird.	

Beispiel

```
scandisk A: /A
```

Das Dateisystem des Diskettenlaufwerks A: wird überprüft. Fehlerkorrekturen erfolgen automatisch.

B.28 SESSION

Funktion Eine Vordergrundtask auf einem anderen Terminal starten

Inline-Kommando

Syntax session <Device> <Unit> <Command>

Parametername	Bedeutung
`Device`	Gerätetreiber-Nummer des Terminaltreibers (normalerweise der BYT-Treiber)
`Unit`	Geräte-Nummer des zu verwendenden Terminals
`Command`	CLI-Kommando, das als Vordergrundtask ausgeführt werden soll

Beschreibung

`SESSION` startet einen Vordergrund-Job, auf dessen Beendigung nicht gewartet wird. Er unterscheidet sich von einem Hintergrund-Job durch die Verwendung eines Terminals. Als <`Command`> kann jedes nachladbare CLI-Kommando oder Programm verwendet werden. Werden beim Aufruf von `SESSION` nicht alle Parameter angegeben, werden die fehlenden Parameter interaktiv abgefragt.

Als <`Command`> können Sie außerdem das Pseudo-Kommando CLI verwenden, das die CLI-Task startet, also eine neue CLI-Session startet.

Die C-Streams `stdin`, `stdout` und `stderr` für den gestartete Vordergrund-Job werden auf das angegebene Terminal gesetzt.

Beispiel

```
SESSION 0 1 HSFSTEST
```

Startet das Programm HSFSTEST (nicht Teil des CLI) als interaktives Programm auf Device 0 Unit 1.

```
SESSION 0 1 CLI
```

Startet eine interaktive CLI-Session auf Device 1 Unit 0.

```
SESSION 0 1 CLI C:\CLI\CLISTART.BAT
```

Startet eine interaktive CLI-Session auf Device 1 Unit 0. Die Datei CLISTART.BAT wird beim Starten der CLI-Task ausgeführt.

B.29 SET

Funktion Environment-String definieren

Inline-Kommando

Syntax set [<String_1>=[<String_2>]]

Beschreibung

Definiert einen Environment-String mit der CRUN-Funktion `putenv`. Falls `<String_2>` fehlt, wird `<String_1>` aus dem Environment entfernt.

Wird nur `SET` eingegeben, werden alle aktuellen Environment-Strings ausgegeben.

Bemerkungen

Für jede vom CRUN verwaltete Task wird ein eigenes Environment geführt. Sie können Environment-Strings in CLI-Kommandos verwenden, indem Sie den Namen in %-Zeichen einklammern, z.B. %TEMPDIR%. Sie können auf Environment-Strings in Ihren Programmen zugreifen, indem Sie die von der CRUN unterstützte ANSI-C Funktion `getenv` verwenden.

Beispiel

```
SET TEMPDIR=C:\TEMP
```

Setzt den Environment-String TEMPDIR gleich C:\TEMP.

```
SET TEMPDIR=
```

Entfernt den Environment-String TEMPDIR vom Environment.

```
SET
```

Listet alle aktuellen Environment-Strings auf.

B.30 START

Funktion Einen Hintergrund-Job starten

Inline-Kommando

Syntax start [/N] [/P=nn] <Kommando>

Parametername	Bedeutung
N	gibt an, daß ein numerischer Coprozessor verwendet wird
P=nn	gibt die Task-Priorität zwischen 0 und 255 an
Kommando	gültiges nachladbares CLI-Kommando, Programm oder ein Batchdatei

Beschreibung

Mit START wird eine Hintergrund-Job gestartet. `stdin`, `stdout` und `stderr` können mit den Redirektionszeichen '<' und '>' umgeleitet werden, die sich sowohl auf START als auch auf den gestarteten Hintergrund-Job beziehen. Falls nicht explizit angegeben, werden `stdin`, `stdout` und `stderr` jeweils auf die NUL-Datei eingestellt. Falls keine Priorität angegeben wird, erhält der Hintergrund-Job die um 1 dekrementierte Priorität der CLI-Session.

Dem Hintergrund-Job wird eine Job-Nummer zugewiesen, die bei `CANCEL` und `CPRI` verwendet werden kann. Falls ein Batchdatei als <Kommando> angegeben wird, wird eine neue CLI-Task als Hintergrund-Job gestartet, um die Batchdatei auszuführen.

Ein Hintergrund-Job hat definitionsgemäß keine Konsole. Falls Sie einen Job starten möchten, der Ausgaben auf eine Konsole macht, aber auf dessen Beendigung nicht gewartet wird, müssen Sie das Kommando SESSION verwenden.

Die Vordergrund-Job setzt sich sofort nach dem Starten des Hintergrund-Jobs fort. Falls ein Fehler beim Aufbau der Programmumgebung des Hintergrund-Jobs vorkommt, kann es passieren, daß sich der Hintergrund-Job ohne jegliche Meldung beendet. Dies kommt beispielsweise vor, wenn `stdout` in eine Datei umgeleitet werden soll, die Datei aber nicht geöffnet bzw. angelegt werden kann.

Beispiel

```
START /P=10 COPY A:\*.* C:\
```

B.31 SYSTAT

Funktion Aktive CLI-Jobs anzeigen

Inline-Kommando

Syntax systat

Beschreibung

Alle aktive CLI-Jobs werden im folgenden Format angezeigt:

```
Tsk Pjob Job  Pri Command              stdout          Start time
1Eh      0    64  CLI C:\CLISTART.BAT  Device=0 Unit=0 31-MAY-1992 09:14:06
1Eh 0    3    64  systat               Device=0 Unit=0 31-MAY-1992 09:29:38
22h      1    64  CLI C:\CLISTART.BAT  Device=0 Unit=1 31-MAY-1992 09:14:12
26h 1    2    64  dir c: /s            Device=0 Unit=1 31-MAY-1992 09:29:37
```

Die einzelnen Spalten haben folgende Bedeutung:

Tsk	Die Task-ID (wird beim Starten der Task vergeben)
Pjob	Die Nummer des Vaterjobs
Job	Die Jobnummer
Pri	Die Priorität des Jobs
Command	Das Kommando oder die Task selbst
stdout	Die Ausgabeeinheit/Datei des Jobs
Start time	Datum und Zeit vom Starten des Jobs

Inline-Kommandos, nachladbare Kommandos, CLI-Sessions und jede vom CLI gestartete Task werden aufgelistet. Die CLI-Session wird mit dem Pseudo-Kommando `CLI <Batchfile>` aufgeführt.

Beispiel

```
SYSTAT
```

B.32 TIME

Funktion Die aktuelle Uhrzeit anzeigen oder ändern

nachladbares Kommando

Syntax time [<time>]

Parametername	Bedeutung
time	Neue Uhrzeit im Format <Stunden>:<Minuten>:<Sekunden>.

Beschreibung

Falls TIME ohne Parameter eingegeben wird, wird die aktuelle Uhrzeit angezeigt und darauf gewartet, daß der Benutzer eine neue Uhrzeit eingibt. Durch das einfache Drücken der Eingabetaste bleibt die Uhrzeit unverändert. Sie können die Sekunden, oder Minuten und Sekunden weglassen. Die Defaultwerte sind jeweils 0.

Beispiel

```
TIME
Current time 16:21:03
Enter new time:

TIME 16:41
```

B.33 VER

Funktion M7 RMOS32 Systeminformationen ausgeben

Inline-Kommando

Syntax ver [*/v* | */d*]

Parametername	Bedeutung
/v	Zusätzliche Information ausgeben
/d	Debug-Information ausgeben

Beschreibung

Das Kommando gibt Systeminformationen wie folgt aus:

Ohne Angabe von Parametern wird die aktuelle Version ausgegeben:

```
RMOS3 Version  4.00.00
```

Mit dem Parameter */v* liefert das Kommando weitere Informationen über das System. Die Ausgabe hat folgenden Aufbau:

```
RMOS3 Version  4.00.00
running the file C:\RM3_PC1.SYS on a Pentium processor, FPU present
```

Mit dem Parameter */d* liefert das Kommando noch detailliertere Debug-Informationen. Die Ausgabe hat folgenden Aufbau:

```
RMOS3 Version  4.00.00
running the file C:\RM3_PC1.SYS on a Pentium processor, FPU present
flat code selector is 0x60, flat data selector is 0x64
systemrate is 10 msec
46 SMRs of 80 are available (SMR out of bound reached 0 times)

there are 2 interrupt controllers available
PIC Nr.        Base           Mask
 1             0x60           0x20
 2             0x70           0x00
```

B.34 CLI-Fehlermeldungen

Die folgenden Fehlercodes werden als Exitstatus von allen CLI-Kommandos verwendet.

Die Bedeutung eines Errorcodes kann mit dem CLI-Kommando ERROR angezeigt werden. Die Werte sind in der Datei CLI.H definiert.

Fehler-Nummer	Fehlermeldung	Bedeutung
0	E_CLI_OK	Kein Fehler
0x8201	E_CLI_NOT_RESIDENT	Keine residente Task
0x8202	E_CLI_FILE_FORMAT	Ungültiges Dateiformat
0x8203	E_CLI_DYNAMIC_MEM	Dynamischer Speicher nicht ausreichend
0x8204	E_CLI_INTERNAL_ERROR	Interner Fehler
0x8205	E_CLI_LTT_FULL	Loaded Task Tabelle voll
0x8206	E_CLI_CJT_FULL	CLI Job Tabelle voll
0x8207	E_CLI_CREATE_ERROR	Fehler beim Erzeugen einer Task
0x8208	E_CLI_START_ERROR	Fehler beim Starten einer Task
0x8209	E_CLI_DELTSK_ERROR	Fehler beim Löschen einer Task
0x820A	E_CLI_LT_ACTIVE	Die angegebene geladene Task ist rechnend
0x820B	E_CLI_JOB_ACTIVE	Der angegebene CLI Job ist rechnend
0x820C	E_CLI_FILE_NOT_FOUND	Datei nicht gefunden
0x820D	E_CLI_NOT_A_CLI_JOB	Die Task ist kein CLI Job
0x820E	E_CLI_NOT_A_CRUN_JOB	Die Task ist keine CRUN Task
0x820F	E_CLI_NOT_INLINE	Kein Inline Kommando
0x8210	E_CLI_SYNTAX_ERROR	Syntaxfehler
0x8211	E_CLI_ENVIRONMENT_FULL	Kein Speicher für neues Environment vorhanden
0x8212	E_CLI_NO_COMMAND	Kommando nicht verfügbar
0x8213	E_CLI_EOF	Dateiende
0x8214	E_CLI_EXIT_STATUS	Exit-Status (als Rückgabewert des EXIT-Kommandos)
0x8215	E_CLI_CATALOG_FULL	RMOS Catalog voll
0x8216	E_CLI_CREATESEMA_ERROR	Fehler beim Anlegen eines Semaphors
0x8217	E_CLI_REDIR_STDIN_FAIL	Fehler beim Umlenken von stdin
0x8218	E_CLI_REDIR_STDOUT_FAIL	Fehler beim Umlenken von stdout
0x8219	E_CLI_REDIR_STDERR_FAIL	Fehler beim Umlenken von stderr
0x821A	E_CLI_UNKNOWN_SWITCH	Unbekannter Schalter
0x821B	E_CLI_TOO_MANY_PARAMS	Zu viele Parameter
0x821C	E_CLI_INVALID_FILENAME	Dateiname ungültig
0x821D	E_CLI_INVALID_LT_INDEX	Ungültiger Taskindex
0x821E	E_CLI_INVALID_JOB_INDEX	Ungültige CLI-Job Nummer
0x821F	E_CLI_TOO_FEW_PARAMS	Zuwenig Parameter angegeben
0x8220	E_CLI_FILE_READ_ERROR	Fehler beim Lesen einer Datei

Fehler-Nummer	Fehlermeldung	Bedeutung
0x8221	E_CLI_FILE_WRITE_ERROR	Fehler beim Schreiben einer Datei
0x8222	E_CLI_COPY_TO_SELF	Versuch eine Datei auf sich selbst zu kopieren
0x8223	E_CLI_UNEXPECTED_EOF	Unerwartet Dateiende erreicht
0x8224	E_CLI_PERMANENT_TASK	Permanente CLI-Task
0x8225	E_CLI_NO_PROMPT	Fehlende Promptdefinition
0x8226	E_CLI_JOB_ABORTED	Der Job wurde abgebrochen
0x8227	E_CLI_NOT_A_DEVICE	Kein Gerätename
0x8228	E_CLI_MKDIR_FAIL	Anlegen eines Dateiverzeichnisses fehlgeschlagen
0x8229	E_CLI_BATCH_FILE	Datei ist eine Batch-Datei
0x822A	E_CLI_NOT_MOUNTED	Datenträger nicht angemeldet
0x822B	E_CLI_INVALID_WILDCARD	Gebrauch von Wildcards nicht korrekt
0x822C	E_CLI_NOT_A_DIRECTORY	Kein Dateiverzeichnis
0x822D	E_CLI_DIR_OPEN_FAIL	Öffnen des Dateiverzeichnisses fehlgeschlagen
0x822E	E_CLI_FILE_OPEN_FAIL	Öffnen einer Datei fehlgeschlagen
0x822F	E_CLI_FILE_CLOSE_FAIL	Schließen einer Datei fehlgeschlagen
0x8230	E_CLI_SVC_ERROR	SVC-Fehler
0x8231	E_CLI_DEV_READ_ERROR	Fehler beim Lesen von einem Gerät
0x8232	E_CLI_DEV_WRITE_ERROR	Fehler beim Schreiben auf ein Gerät
0x8233	E_CLI_IS_A_DIRECTORY	Die Datei ist ein Verzeichnis
0x8234	E_CLI_INLINE	Inline CLI-Kommando
0x8235	E_CLI_COMPLETED	CLI Job ist beendet
0x8236	E_CLI_UNINITIALISED	CLI Job wurde nicht initialisiert
0x8237	E_CLI_OUT_OF_RANGE	Index out of range
0x8238	E_CLI_NO_SPACE_FOR_NPX	Beim Anlegen der Task konnte kein NPX Kontext eingerichtet werden
0x8239	E_CLI_NO_PARENT_JOB	Kein Vater-Job
0x823A	E_CLI_INVALID_VOL_NAME	Datenträgername ungülig
0x823B	E_CLI_MOUNTED	Datenträger ist gemountet
0x823C	E_CLI_INVALID_DATE_TIME	Datum- oder Zeitformat ungültig
0x823D	E_CLI_READ_ONLY	Datei ist schreibgeschützt

Rückgabewerte des Task-Laders

In der folgenden Tabelle werden die Rückgabewerte der Task-Lader-Funktionen `stl_load` und `stl_free` und ihre Bedeutung aufgeführt. Die Fehlernummern können beim Laden von Tasks durch den CLI auftreten.

Fehler-Nummer	Fehlermeldung	Bedeutung
0H	**E_LD_OK**	Das Laden war erfolgreich.
8100H	**E_LD_NO_REG_INIT**	Missing register initialization Initial-Registerwerte sind nicht vorhanden (d.h. keine Startadresse). Das Programm wurde trotzdem geladen.
8101H	**E_LD_NO_FIXUPS**	Missing fixup initialization Ein STL- oder PE-Modul, das keine Relocation-Information enhält, wurde geladen. Beim STL-Modul werden die Relocation-Informationen von BND386 mit der Option RCONFIGURE erzeugt. Sie ist in einem STL-Modul nur bedingt vorhanden, da sie bei anderen Betriebssystemen nicht immer notwendig ist. Unter Umständen ist diese Information auch bei RMOS nicht notwendig, z.B., wenn ein Programm nur ein Segment besitzt.
8110H	**E_LD_FILE_OPEN_FAIL**	Cannot open file Die Programmdatei konnte nicht geöffnet werden. Vermutlich existiert sie nicht.
8111H	**E_LD_FILE_FORMAT**	Unkown file format Die Datei hat ein unbekanntes Format. Das Format wird wie folgt am ersten Byte oder Wort der Datei erkannt: STL: 1. Wort = 0206H PE: 1. 16-Bit-Wort = ”M2” 1. 32-Bit Wort of NEW EXE Header = 00004550H, Magic number von optionalem Header == 0413
8112H	**E_LD_UNEXPECTED_EOF**	Unexpected end of file Bei einem Lesezugriff auf die Datei wurden keine oder zu wenig Bytes übertragen.
8113H	**E_LD_DYNAMIC_MEMORY**	Insufficient memory Es konnte kein oder nicht ausreichend Speicher alloziert werden.
8114H	**E_LD_ABSOLUTE**	Module contains absolute segments Das Modul enthält ein oder mehrere Segmente, die an feste Adressen gebunden wurden oder ist eine PE-Systemdatei.
8115H	**E_LD_DALOC**	Loader result segment destroyed Ein Fehler ist beim Aufruf von `stl_free` aufgetreten. Das Loader-Result-Segment wurde überschrieben.
8116H	**E_LD_NULL_POINTER**	Null pointer `stl_free` wurde mit einem Pointer auf das Loader-Result-Segment aufgerufen, der gleich Null ist.

Fehler-Nummer	Fehlermeldung	Bedeutung
8117H	**E_LD_OVERLAY**	Module contains overlays Das Modul enthält eine Overlay-Beschreibung.
8118H	**E_LD_INVALID_TASK_ OR_DRIVER**	Not a RMOS application/driver Das Programm ist keine M7 RMOS32–Anwendung
8130H	**E_LD_PROCESSOR_TYPE**	Invalid processor type Der im Modul-Header angegebene Prozessortyp ist nicht gleich 80386, 80486 oder Pentium.
8131H	**E_LD_NON_EXECUTABLE**	Module not executable Das Modul hat nicht das Attribut ”executable”, das im Modul-Header angegeben wird. Das bedeutet, daß es beim Binden undefinierte Symbole gegeben hat. Beim PE-Format kann es auch bedeuten, daß das Programm entrelativiert ist oder versucht wurde ein Library Image zu laden.
8132H	**E_LD_MULTI_LDT**	Module contains multiple LDTs Das Modul enthält mehrere Local-Descriptor-Tables (LDT’s)
8133H	**E_LD_GATES_PRESENT**	Gates present Der DESCRP-Record enthält Gate-Descriptoren.
8134H	**E_LD_NO_STACK_DESC**	No stack segment Es ist kein Stacksegment definiert.
8135H	**E_LD_NULL_SEG_FIXUP**	Null segment fixup Das Modul enthält einen Fixup zu einem Zielsegment, dessen Selector kein Code- oder Daten-Segment entspricht.
8136H	**E_LD_BAD_SEG_FIXUP**	Invalid segment fixup Das Modul enthält einen Fixup zu einem Zielsegment, dessen Selector ungültig ist.
8137H	**E_LD_ACCESS_RIGHTS**	Access rights error Ein Fehler ist beim Aufruf von `RmChangeDescriptorAccess` aufgetreten.
8138H	**E_LD_ITERATE**	Iterate data segment has iBSS format Ein ”Iterated Data Segment” besitzt das Attribut iBSS.
8140H	**E_LD_FIXUP_FORMAT**	Invalid fixup Ein Fixup hat ein illegales oder nicht unterstütztes Format.
8141H	**E_LD_USE16**	USE16 segment found Ein Segment hat das USE16-Attribut. Bei RMOS3 werden nur Segmente mit dem Attribut USE32 unterstützt.
8150H	**E_LD_SEEK**	File seek error Die Dateiposition konnte nicht entsprechend gesetzt werden.
8151H	**E_LD_DLL**	Module is a DLL Die Datei ist eine Dynamic Link Library.
8152H	**E_LD_INVALID_ DESCRIPTOR**	Invalid descriptor Beim Konvertieren einer Segmentadresse zu einer physikalischen ist ein Fehler aufgetreten.
8153H	**E_LD_TOO_MANY_OBJECTS**	Too many objects

Fehler-Nummer	Fehlermeldung	Bedeutung
8158H	**E_LD_OUT_OF_SEMAPHO-RES**	Couldn't create a semaphore Das für den Task–Lader notwendige Semaphor STL_SEMA konnte nicht erzeugt werden
8159H	**E_LD_CATALOG_EXCEEDED**	Couldn't catalog semaphore Das für den Task–Lader notwendige Semaphor STL_SEMA konnte nicht katalogisiert werden

Debugger-Kommandos

C

Übersicht

In diesem Kapitel finden Sie die Kommandos des Low-Level-Debuggers in alphabetischer Reihenfolge zum Nachschlagen.

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
C.1	Syntax der Debugger-Kommandos	C-3
C.2	ASM	C-7
C.3	BASE	C-9
C.4	BREAKS	C-10
C.5	CALCULATE	C-11
C.6	CALL	C-12
C.7	CHANGE	C-13
C.8	CONT	C-14
C.9	CPUREG	C-15
C.10	DIR	C-16
C.11	DISPLAY	C-19
C.12	EVALUATE	C-20
C.13	EXIT	C-21
C.14	EXITK	C-22
C.15	FILL	C-23
C.16	FREETASK	C-24
C.17	GO	C-25
C.18	HALT	C-26
C.19	HELP	C-27
C.20	IN	C-28
C.21	INHIB	C-29
C.22	KILL	C-30
C.23	LINES	C-31
C.24	LOADTASK	C-32

Im Kapitel	finden Sie	auf Seite
C.25	MONITOR	C-34
C.26	OUT	C-35
C.27	QUALIFY	C-36
C.28	QUERY	C-37
C.29	REGS	C-39
C.30	REPORT	C-41
C.31	SET	C-47
C.32	STACK	C-49
C.33	START	C-50
C.34	STEP	C-51
C.35	SVC	C-52
C.36	SWITCH	C-54
C.37	TASK	C-55
C.38	TCB	C-56
C.39	TCD	C-57
C.40	Fehlermeldungen des Debuggers	C-58

C.1 Syntax der Debugger-Kommandos

Allgemein hat ein Debugger-Kommando die Form:

```
<wiederholfaktor><leer>]<Kommando>[;[<wiederholfaktor>
<leer>]<Kommando>]...<return>
```

wobei

```
<Kommando> =<Kommandowort><DEL><Arg1><DEL><Arg2>...<Return>
```

Werden zwingend erforderliche Argumente weggelassen, so fordert der Debugger deren Eingabe in einer neuen Zeile an und gibt in der Eingabeauffordeung Hinwise über die Art der erwarteten Argumente.

Nachfolgend werden die häufig verwendeten Argumente beschrieben.

Adresse

<address>	{<selector>:<offset>|<abs_addr>{L|P}}
<selector>	Selektor, der einen Deskriptor in der aktuellen GDT bzw. LDT referenziert. <selector> kann <expr> sein
<offset>	32-bit-Wert, <offset> kann <expr> sein
<abs_addr>	ist eine absolute 32-Bit-Adresse
L/P	Schlüsselwort für lineare/physikalische Adresse (z.Z. nur lineare Adresse, auch wenn P angegeben)
<expr>	(siehe unten)

Für die Angabe eines Selektors können die Segmentregister verwendet werden. Die Task-Registernamen stehen jedoch nur im Breakpoint-Kontext zur Verfügung. Ein Selektor muß sich immer auf einen gültigen Segmentdescriptor beziehen (ER/RW). Ist dies nicht der Fall, wird die Fehlermeldung

```
WRONG SELECTOR
```

ausgegeben. Die Eingabe eines Selektors gilt als Voreinstellung für nachfolgende Kommandos, wenn in diesen für die Adresse nur ein Offset-Wert angegeben wird.

Auch eine katalogisierte Adresse kann in einfachen Hochkommata eingegeben werden, wenn sie als Typ MISC oder SYSTEM im Katalog eingetragen ist.
ID=selector (wenn =0, dann IDE = lineare Adresse)
IDE=offset
Dies gilt für die Kommandos ASM, CALL, CHANG, DISPL, FILL und SET und nicht für: CALC, EVAL, GO und START.

Ausdruck <expr>

Ein mit `<expr>` bezeichneter Ausdruck kann enthalten:
Konstanten, Registerinhalte, Speicherinhalte, Eingabe-Ports, Verknüpfungen mit arithmetischen und logischen Operatoren:

<expr>	[{-|~}] <term> [[{+|-|^|'|'|&|<<|>>} <term>]...]
<term>	<factor>[[{*|/|%}<factor>]...]
<factor>	{<ident>|<const>|(<expr>)}
<ident>	{<cpureg>|<memory>|<port>}
<cpureg>	{CS|DS|ES|SS|FS|GS|EIP|EAX|EBX|ECX|EDX|EDI|ESI| ESP|EBP|EFL|IP|AX|BX|CX|DX|DI|SI|BP|FL|AL|AH|BL|BH |CL|CH|DL|DH|GDTR|LDTR|IDTR|TR|CR0|CR2| CR4**1)**|DR0|DR1|DR2|DR3|DR6|DR7|TR1**2)**|TR2**2)**|TR3**1)**| TR4**1)**|TR5**1)**|TR6|TR7|TR9**2)**|TR10**2)**|TR11**2)**|TR12**2)**} **1)** nur ab Prozessor 486 mit CPUID **2)** nur ab Pentium-Prozessor
<memory>	{BYTE|WORD|DWORD|POI|SPOI}'['<address>']' <port> {INB|INW|IND} '['<port_address>']'
<port_address>	<expr>
~	NOT
-	Negation
+ - * /	arithmetische Grundoperationen
%	Rest bei Integerdivision
^	XOR
|	OR
&	AND
<<	links schieben
>>	rechts schieben

Konstante (const)

Konstanten können in hexadezimaler, dezimaler oder oktaler Zahlenbasis, ebenso in ASCII Notation eingegeben werden. Konstanten werden außerdem nach ihrem Speicherformat unterschieden:

Byte, Word, Dword, Qword

Begrenzer (DEL)

Folgende Zeichen sind als Begrenzer erlaubt:

- Leerzeichen ( )
- Komma (,)
- Gleichheitszeichen (=)

Es sind mehrere Begrenzer (DEL) zwischen zwei aufeinanderfolgenden Argumenten erlaubt.

Format

<format> {[<memory_format>][<display_format>]|data_type}

<memory_format> {byte|word|dword|qword}

wählt die Elementlänge eines spezifizierten Speicherbereichs

byte	1 Byte
word	2 Byte
dword	4 Byte
qword	8 Byte

<display_format> {Mask|Octal|Hex|Unsigned|Signed|Ascii}

wählt das Darstellungsformat auf der Konsole

Mask	Darstellung als Bitmuster: xxxx xxxx
Octal	Darstellung als Oktalzahl
Unsigned	Darstellung als vorzeichenlose Dezimalzahl
Signed	Darstellung als vorzeichenbehaftete Dezimalzahl
Hex	Darstellung als Hexadezimalzahl
Ascii	Darstellung als Zeichenstring (1 Byte pro Buchstabe)

<data_type> {Short|Long|Float|Double|Tempreal|Pointer|Spointer}

(beinhaltet memory_format + display_format)

Short	Signed Word
Long	Signed Dword
Pointer	(Long Pointer) 16-Bit-Selektor:32-Bit-Segment
Spointer	(Short Pointer) 16-Bit-Selektor:16-Bit-Segment
Float	4-Byte-Realzahl
Double	8-Byte-Realzahl
Tempreal	10-Byte-Realzahl

Task-ID

<task_id> <expr>

Gültige Werte für <task_id> sind 0 - 2047 sowie 0FFFFH.
Auch eine katalogisierte Adresse kann in einfachen Hochkommata eingegben werden (überall, mit Ausnahme von quality und report task).

Breakpoints

Beim Setzen von Haltepunkten wird unterschieden zwischen Hard-Breakpoints, Soft-Breakpoints und Debugregister-Breakpoints.
Hard- und Soft-Breakpoints ersetzen den entsprechenden OP-Code mit der INT3-Anweisung und können nur verwendet werden, wenn sich der Programmcode im RAM-Speicher befindet. Soft-Breakpoints sind nur temporär vorhanden, d.h. sie werden automatisch nach der Abarbeitung gelöscht. Im Gegensatz dazu bleiben Hard-Breakpoints erhalten, bis sie mit `kill` entfernt werden.

Debugregister-Breakpoints stehen beim 80386-/80486 und Pentium-Prozessor zur Verfügung, wo sie zum Setzen von Haltepunkten für Schreib-/Lese-Speicherzugriffe verwendet werden. Der Prozessor erlaubt außerdem das Testen mit Execution-Breakpoints für Programmcode, der sich im EPROM-Speicher befindet.

<breakpoint>{$|@|E|{M|W}{B|W|D}}1...0F

$	Soft-Breakpoint setzen
@	Hard-Breakpoint setzen

Breakpoint im Debugregister setzen:

E	Execution-Breakpoint im Debugregister
MB	Modify (Read or Write) Byte Breakpoint
MW	Modify (Read or Write) Word Breakpoint
MD	Modify (Read or Write) Dword Breakpoint
WB	Write Byte Breakpoint
WW	Write Word Breakpoint
WD	Write Dword Breakpoint

C.2 ASM

Funktion **Speicherinhalt disassemblieren**

Syntax **ASM <address> [{LEN <expr> | TO <expr>}]**

Beschreibung

ASM disassembliert den Speicher ab <address>, wobei der Inhalt als 8086/80186/80286/80386/80486 oder Pentium bzw. 8087/80287/80387 Code interpretiert wird.

Wird das Schlüsselwort TO eingegeben, bezeichnet <expr> den Offset-Wert für das Ende des zu disassemblierenden Speicherbereichs. Mit LEN <expr> wird die Anzahl der Befehlssequenzen direkt angegeben. Ohne Längenangabe werden 10 Zeilen ausgegeben.

Die Ausgabe kann per Tastendruck (<Ctrl>+S, <Ctrl>+Q, <Ctrl>+C) zeitweise angehalten bzw. abgebrochen werden.

Hinweis

Anmerkung zum Protected Mode des 80386/80486:

Wird <address> über einen Selektor spezifiziert, muß der dazugehörige Deskriptor vom Typ Codesegment sein. Das Defaultbit im Codesegment-Deskriptor bestimmt den Disassembliermodus (USE32/USE16) und damit die Bedeutung der Operanden- und Adreß-Präfixe. Wird eine lineare/physikalische Adresse angegeben, wird immer USE 32 disassembliert. Die Ausgabe der Befehlsadressen erfolgt in Abhängigkeit der Adreßangabe bei der Kommandoeingabe in der Form ”Lineare Adresse” bzw. ”Selektor:Offset”. Im **Breakpoint-Kontext** kann mit der Adreßangabe CS:EIP der dem Breakpoint folgende Codebereich disassembliert werden.

Beispiel

Erste 20 Anweisungen von Task mit ID=2 disassemblieren. Der Codebeginn ist aus dem TCD-Block ersichtlich.

```
DEBUG_T>TCD 2<CR>
Task entry point: 005C:00008A5F, TCD address= 0074:000406B,
DS = 0000, ES = 0000, Stack base = 0084:00002308

DEBUG_T>ASM 5C:8A5F LEN 20<CR>
005C:00008A5F   A0D2010000       MOV    AL,BYTE PTR  [01D2]
005C:00008A64   08C0             OR     AL,AL
005C:00008A66   0F844B020000     JZ     08CB7     ; (= $+0251)
005C:00008A6C   3C12             CMP    AL,12
005C:00008A6E   750C             JNZ    08A7C     ; (= $+0E)
005C:00008A70   C605D003000001   MOV    BYTE PTR [03D0],01
005C:00008A77   E9BF000000       JMP    08B3B     ; (= $+0C5)
005C:00008A7C   0FB605D2010000   MOVZX  EAX,BYTE PTR [01D2]
005C:00008A83   6BC80F           IMUL   ECX,EAX,0F
005C:00008A86   0FB799AC020000   MOVZX  EBX,WORD PTR [ECX+02AC]
005C:00008A8D   81FBFFFF0000     CMP    EBX,0FFFF
005C:00008A93   7409             JZ     08A9E     ; (= $+0B)
005C:00008A95   663B1DD6010000   CMP    BX,WORD PTR [01D6]
005C:00008A9C   752D             JNZ    08ACB     ; (= $+2F)
005C:00008A9E   0FB605D2010000   MOVZX  EAX,BYTE PTR [01D2]
005C:00008AA5   6BD00F           IMUL   EDX,EAX,0F
005C:00008AA8   8A9AA3020000     MOV    BL,BYTE PTR [EDX+02A3]
005C:00008AAE   3A1DD4010000     CMP    BL,BYTE PTR [01D4]
005C:00008AB4   7515             JNZ    08ACB     ; (= $+17)
005C:00008AB6   668B9AAE020000   MOV    BX,WORD PTR [EDX+02AE]
005C:00008ABD   664B             DEC    BX
005C:00008ABF   66899AAE020000   MOV    WORD PTR [EDX+02AE],BX
005C:00008AC6   6609DB           OR     BX,BX
005C:00008AC9   742D             JZ     08AF8      ; (= $+2F)
005C:00008ACB   C6050402000019   MOV    BYTE PTR [0204],19
005C:00008AD2   C6050502000004   MOV    BYTE PTR [0205],04
005C:00008AD9   66A1D6010000     MOV    AX,WORD PTR [01D6]
005C:00008ADF   66A306020000     MOV    WORD PTR [0206],AX
005C:00008AE5   B804020000       MOV    EAX,0204
005C:00008AEA   1E               PUSH   DS
005C:00008AEB   50               PUSH   EAX
005C:00008AEC   E887C40000       CALL   014F78    ; (= $+0C48D)
```

C.3 BASE

Funktion **Zahlenbasis einstellen**

Syntax **BASE [{10|16}]**

Beschreibung

BASE ohne Parameterangabe zeigt die aktuell eingestellte Zahlenbasis an. BASE=10 schaltet auf dezimale Zahleninterpretation, BASE=16 auf hexadezimale Interpretation. Das mit BASE eingestellte Format kann durch die folgenden Präfixe für einzelne Zahleneingaben umgestellt werden:

0x	hexadezimal
0n	dezimal
0o	oktal
0m	dual
”	ASCII

C.4 BREAKS

Funktion **Breakpoint(s) auflisten**

Syntax **BREAKS**

Beschreibung

BREAKS listet alle derzeit gesetzten Breakpoints mit der zugehörigen Adresse auf. Die Adresse hat das Standard-Format Segment:Offset bzw. Selektor:Offset oder lineare Adresse (siehe Kommando SET). Die Ausgabe erfolgt hexadezimal.

Beispiel

siehe QUALIFY

C.5 CALCULATE

Funktion

Gleitpunkt-Ausdruck berechnen

Syntax

CALCULATE <real_expr>

```
<real_expr>  [-] <term> [[{+|-} <term>]...]
<term>       <factor> [[{*|/} <factor>]...]
<factor>     {<ident>|<const>|(<expr>)}
<ident>      {<32bitreg>|<Speicher>}
<32bitreg>   {EAX|EBX|ECX|EDX|EDI|ESI|ESP|EBP}
<Speicher>   {Float|Double|Tempreal} '['<address>']'

-            Negation
+ - * /      arithmetische Grundoperationen
```

Syntax für die Eingabe von Gleitkomma-Zahlen:

```
<real_const>
[{+|-}]<digt>['.'[[<digit>]...][E[{+|-}][[<digit>]...]]
<digit>      {0|1|2|3|4|5|6|7|8|9}(Nuc1/2:18.7.9.2)
```

Beschreibung

Der angegebene Ausdruck wird berechnet und das Ergebnis an der Konsole angezeigt. Die Berechnung und die Ausgabe erfolgen immer im Longreal-Format.
Für die Exponentenangabe ist nur E zulässig. Bei der Ausgabe wird je nach Format unterschieden: E, D, T (Float, Double, Tempreal).

Beispiel

```
DEBUG_T>CALC   3.141592654E+5 * 2.718281828E-3
                +8.53973422234649D+002
```

C.6 CALL

Funktion

Ausführen eines Programms

Syntax

CALL <address> [<stacksize>|[<datasize>]]

`<stacksize> = <expr.>`	Länge des Stacksegments in Bytes, max. 64 KByte, Default-Wert 400
`<datasize> = <expr.>`	Länge des Datensegments in Byte, max. 64 KByte, Default-Wert 100 Bytes

Beschreibung

Ausführen eines Anwenderprogramms, das vorher in den Speicher geladen wurde.

Der Debugger kreiert und startet eine Task, die einen Aufruf (Far Call) auf die angegebene Adresse (Startparameter) durchführt. Die Task wird mit dem String ”CALL_XXXX” (XXXX=Task-ID) in den Betriebsmittelkatalog eingetragen. Bei einer linearen/physikalischen Adreßangabe wird ein entsprechender Descriptor erzeugt (ER,USE32). Der Debugger wartet beim Aufruf des Kommandos nicht auf die Beendigung der Task, so daß mehrere Tasks parallel gestartet werden können.

Nach Beendigung des Anwenderprogramms wird der Betriebsmittel-Eintrag gelöscht und der SVC `delete` abgesetzt. Anschließend wird ebenfalls die Task wieder gelöscht. Für das Datensegment und das Stacksegment wird standardmäßig ein Speicherbereich wie oben angegeben allociert. Das DS-, ES-, SS-Register im TCD wird entsprechend vorbesetzt.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

C.7 CHANGE

Funktion **Speicherbytes anzeigen/verändern**

Syntax **CHANGE [<format>] <address> [<expr>[,<expr>]...]**

Beschreibung

Der ab Adresse <address> angegebene Speicherbereich wird mit den neuen Werten <expr> überschrieben. Es werden soviele Elemente überschrieben, wie Werte angegeben sind. Entfällt die Werteliste, zeigt der Debugger den Inhalt von <address> im gewählten Format an; der Inhalt kann dann durch Eingabe eines neuen Wertes geändert werden. Nach der Eingabe von <CR> wird das Kommando mit der nächsten Adresse fortgesetzt. Durch die Eingabe eines ungültigen Zeichens, z.B. Q, wird das Kommando beendet.

Ist die Werteliste ein in Hochkomma eingeschlossener String, wird dieser als ein Wert interpretiert und unabhängig vom gewählten Speicherformat wird jeder Buchstabe in ein Byte geschrieben.

Das Ändern von Speicherinhalten erfolgt ohne Kontroll-Lesen.

CHANGE wird durch Eingabe von q abgebrochen.

Beispiel

Ab Adresse 1A4:4E30 wird der Textstring ABCDEF eingetragen.

```
DEBUG_T>CHANGE WORD 1A4:4E30 4241 4443 4645 <CR>
DEBUG_T>
```

C.8 CONT

Funktion **Angehaltene Task freigeben**

Syntax **CONT <task_id>**

Beschreibung

CONT befreit eine Task mit ID <task_id> aus dem Zustand WARTEND. CONT kann nur auf Tasks angewand werden, die durch den Debugger angehalten bzw. durch einen Breakpoint unterbrochen wurden (Status 26, bzw. Status 2A).

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

Beispiel

```
DEBUG_T>CONT 3 <CR>
DEBUG_T>
```

C.9 CPUREG

Funktion **Registerinhalt prüfen/verändern**

Syntax **<cpureg>[<expr>]**

<cpureg> **{CS|DS|ES|SS|FS|GS|EIP|EAX|EBX|ECX|EDX|EDI|ESI|ESP|EBP|EFL|CR0|IP|AX|BX|CX|DX|DI|SI|BP|FL|AL|AH|BL|BH|CL|CH|DL|DH}**

Beschreibung

Nur im Breakpoint-Kontext:

Der Inhalt von <cpureg> wird angezeigt und kann durch Eingabe eines neuen Werts verändert werden. Ein <CR> beläßt den Registerwert unverändert. Anschließend meldet sich der Debugger mit seinem Prompt.

Beispiel

siehe REGS

C.10 DIR

Funktion **Einträge des Betriebsmittelkatalogs anzeigen**

Syntax **DIR [<SYM_NAM>] [LONG]**

Beschreibung

`DIR` listet alle mit `<SYM_NAM>` bezeichneten Betriebsmittel auf, die im Verzeichnis `CATALOG` verzeichnet sind. Die Ausgabe erscheint in abgekürzter Form, wenn nicht mit `[LONG]` die ausführliche Ausgabe angefordert wird.

Bei fehlender Parameter-Angabe erfolgt die Ausgabe sämtlicher Einträge.

Gültige Parameter sind:

Art des Betriebsmittels	Wert	Symbolischer Name
Task	00H	TASK
Gerätetreiber	01H	DRIV
SpeicherPool	02H	POOL
Semaphore	03H	SEMA
Globales-Ereignis-Flag	04H	FLAG
Überwachter Zugang	05H	CTRL
Lokale Mailbox	06H	LMBX
Reserviert	07H	MISC
Benutzerspezifisch	08H	USER
Diskretes Byte für E/A-Operationen	09H	DSCT
Geräte-Einheit	0AH	UNIT
Message-Queue	0BH	MSGQ

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar

Beispiel

Ausgabe der Betriebsmittel-Einträge

```
>dir
Cataloged resources
Symbolic-name   kind    id          Symbolic-name   kind    id
BU_COUNT        TASK    001CH       BYT             DRIV    0000H
BYT_COM1        UNIT    0004H       BYT_COM2        UNIT    0005H
BYT_EGA_0       UNIT    0000H       BYT_EGA_1       UNIT    0001H
BYT_EGA_2       UNIT    0002H       BYT_EGA_3       UNIT    0003H
CLI_CLEAN_UP    TASK    001AH       CLI_DPAT        TASK    0000H
CLI_JOB_0       TASK    0020H       CLI_MANAGER     SEMA    000BH
CRUN            MISC    007DH       CRUN_GEN        SEMA    000DH
CRUN_HANDLE     SEMA    000EH       CRUN_ID         SEMA    000CH
DEB_0           TASK    0002H       DEB_1           TASK    0003H
DEB_2           TASK    0004H       DEB_3           TASK    0005H
ERRLOG          TASK    0006H       FD0             DRIV    0001H
FD0_0           UNIT    0000H       HD0             DRIV    0002H
HD0_0           UNIT    0000H       HSFS_GLOB       SEMA    000AH
HSF_A           TASK    0008H       HSF_C           TASK    001EH
REMOTE          TASK    0007H       REP             TASK    0001H
RMOSCONF        TASK    0009H       SYSCON          UNIT    0000H

>dir task
Cataloged resources
Symbolic-name   kind    id          Symbolic-name   kind    id
BU_COUNT        TASK    001CH       CLI_CLEAN_UP    TASK    001AH
CLI_DPAT        TASK    0000H       CLI_JOB_0       TASK    0020H
DEB_0           TASK    0002H       DEB_1           TASK    0003H
DEB_2           TASK    0004H       DEB_3           TASK    0005H
ERRLOG          TASK    0006H       HSF_A           TASK    0008H
HSF_C           TASK    001EH       REMOTE          TASK    0007H
REP             TASK    0001H       RMOSCONF        TASK    0009H
```

```
>dir long
Cataloged resources
Symbolic-name kind  id     ide
BU_COUNT      TASK  001CH  FFFFFFFFH
BYT           DRIV  0000H  FFFFFFFFH
BYT_COM1      UNIT  0004H  00010000H   device-id: 00 type: RIO byte
BYT_COM2      UNIT  0005H  00010000H   device-id: 00 type: RIO byte
BYT_EGA_0     UNIT  0000H  00010000H   device-id: 00 type: RIO byte
BYT_EGA_1     UNIT  0001H  00010000H   device-id: 00 type: RIO byte
BYT_EGA_2     UNIT  0002H  00010000H   device-id: 00 type: RIO byte
BYT_EGA_3     UNIT  0003H  00010000H   device-id: 00 type: RIO byte
CLI_CLEAN_UP  TASK  001AH  FFFFFFFFH
CLI_DPAT      TASK  0000H  FFFFFFFFH
CLI_JOB_0     TASK  0020H  FFFFFFFFH
CLI_MANAGER   SEMA  000BH  FFFFFFFFH
CRUN          MISC  007DH  FFFFFFFFH
CRUN_GEN      SEMA  000DH  FFFFFFFFH
CRUN_HANDLE   SEMA  000EH  FFFFFFFFH
CRUN_ID       SEMA  000CH  FFFFFFFFH
DEB_0         TASK  0002H  FFFFFFFFH
DEB_1         TASK  0003H  FFFFFFFFH
DEB_2         TASK  0004H  FFFFFFFFH
DEB_3         TASK  0005H  FFFFFFFFH
ERRLOG        TASK  0006H  FFFFFFFFH
FD0           DRIV  0001H  FFFFFFFFH
FD0_0         UNIT  0000H  00020001H   device-id: 01 type: RIO block
HD0           DRIV  0002H  FFFFFFFFH
HD0_0         UNIT  0000H  00020002H   device-id: 02 type: RIO block
HSFS_GLOB     SEMA  000AH  FFFFFFFFH
HSF_A         TASK  0008H  FFFFFFFFH
HSF_C         TASK  001EH  FFFFFFFFH
REMOTE        TASK  0007H  FFFFFFFFH
REP           TASK  0001H  FFFFFFFFH
RMOSCONF      TASK  0009H  FFFFFFFFH
SYSCON        UNIT  0000H  00010000H   device-id: 00 type: RIO byte

>dir unit long
Cataloged resources
Symbolic-name  kind  id     ide
BYT_COM1       UNIT  0004H  00010000H  device-id: 00 type: RIO byte
BYT_COM2       UNIT  0005H  00010000H  device-id: 00 type: RIO byte
BYT_EGA_0      UNIT  0000H  00010000H  device-id: 00 type: RIO byte
BYT_EGA_1      UNIT  0001H  00010000H  device-id: 00 type: RIO byte
BYT_EGA_2      UNIT  0002H  00010000H  device-id: 00 type: RIO byte
BYT_EGA_3      UNIT  0003H  00010000H  device-id: 00 type: RIO byte
FD0_0          UNIT  0000H  00020001H  device-id: 01 type: RIO block
HD0_0          UNIT  0000H  00020002H  device-id: 02 type: RIO block
SYSCON         UNIT  0000H  00010000H  device-id: 00 type: RIO byte
```

Hinweis

Die symbolischen Bezeichnungen der Betriebsmittel dürfen nicht abgekürzt eingegeben werden.

C.11 DISPLAY

Funktion **Speicherinhalt ausgeben**

Syntax **DISPLAY [<format>] <address> [{LEN|TO} <expr>]**

Beschreibung

Der ab `<address>` spezifizierte Speicherbereich wird entsprechend dem gewählten Format auf der Konsole ausgegeben. Entfällt die Längenangabe wird eine Zeile ausgegeben. Ohne `<format>` wird die Voreinstellung BYTE HEX übernommen. Ist die Adreßangabe ohne Segment/Selektor, wird der zuletzt gewählte Selektor bzw. das zuletzt gewählte Segment benutzt (voreingestellt: Null-Selektor).

Bei byteweiser Ausgabe erscheint zusätzlich eine ASCII-Darstellung. Nicht abbdruckbare Zeichen werden dabei als Punkt dargestellt. Real-Zahlen werden in Exponentenschreibweise dargestellt.

Hinweis

Aus Platzgründen ist die ASCII-Darstellung im Beispiel nicht aufgeführt. Die Ausgabe kann mit <Ctrl>+S oder <Ctrl>+Q angehalten, und mit <Ctrl>+C abgebrochen werden. Die Darstellung der Adresse erfolgt entweder in der Form Selektor:Offset oder als absolute Adresse, je nachdem wie sie in der Kommandozeile eingegeben wurde.

Die Gleitkomma-Datentypen werden bei `DISPLAY`, `FILL` und `CHANGE` nur dann unterstützt, wenn `CALCULATE` konfiguriert ist, da sonst die Module zum Berechnen von Gleitkomma-Ausdrücken nicht eingebunden werden.

Beispiel

Speicherbereich von 100:0 mit Länge 0FH anzeigen

```
DEBUG_T>DISPLAY 100:0 LEN 0F<CR>
0100:0000  xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx xx
DEBUG_T>
```

C.12 EVALUATE

Funktion **Integer-Ausdruck berechnen**

Syntax **EVALUATE <expr>**

Beschreibung

Ein mit <expr> bezeichneter Ausdruck kann enthalten:
Konstanten, Registerinhalte, Speicherinhalte, Eingabe-Ports, Verknüpfungen mit arithmetischen und logischen Operatoren.

```
<expr>          [{-|~}] <term> [[{+|-|^|'|'|&|<<|>>} <term>]...]
<term>          <factor>[[{*|/|%}<factor>]...]
<factor>        {<ident>|<const>|(<expr>)}
<ident>         {<cpureg>|<memory>|<port>}
<cpureg>        {CS|DS|ES|SS|FS|GS|EIP|EAX|EBX|ECX|EDX|EDI|ESI|ESP|
                EBP|EFL|IP|AX|BX|CX|DX|DI|SI|BP|FL|AL|AH|BL|BH|
                CL|CH|DL|DH|GDTR|LDTR|IDTR|TR|CR0|CR2|CR4 1)|DR0|DR1|
                DR2|DR3|DR6|DR7|TR1 2)|TR2 2)|TR3 1)|TR4 1)|TR5 1)|TR6|
                TR7|TR9 2)|TR10 2)|TR11 2)|TR12 2)}
<memory>        {BYTE|WORD|DWORD|POI|SPOI}'['<address>']'
<port>          {INB|INW|IND}'['<port_address>']'
<port_address>              <expr>
```

1) nur ab Prozessor 486 mit CPUID
2) nur ab Pentium-Prozessor

~	NOT
-	Negation
+ - * /	arithmetische Grundoperationen
%	Rest bei Integerdivision
^	XOR
|	OR
&	AND
<<	links schieben
>>	rechts schieben

Der angegebene Ausdruck wird berechnet und das Ergebnis an der Konsole angezeigt. Die Berechnung erfolgt immer im Longinteger-Format. Das Ergebnis wird in den 3 Zahlenbasen (8/10/16) sowie in ASCII-Darstellung ausgegeben.

C.13 EXIT

Funktion **Debugger beenden**

Syntax **EXIT**

Beschreibung

EXIT beendet die Debugger-Task. Beim Verlassen des Debuggers aus dem Breakpoint-Kontext mit EXIT wird die unterbrochene Task nicht freigegeben. Alle gesetzten Breakpoints bleiben erhalten.

Beispiel

```
DEBUG_T>EXIT <CR>
```

C.14 EXITK

Funktion **Breakpoints löschen und unterbrochene Task freigeben**

Syntax **EXITK**

Beschreibung

Alle Breakpoints werden gelöscht. Falls im Breakpoint-Kontext, wird die aktuelle Task fortgesetzt. Anschließend wird die Debugger-Task beendet.
(Wirkungsweise wie bei GO, nur daß vorher alle Breakpoints gelöscht werden.)

Beispiel

```
DEBUG_T>EXITK <CR>
```

C.15 FILL

Funktion **Speicherblock mit Byte-Werten füllen**

Syntax **FILL [<format>] <address> {LEN|TO} <expr>=<expr>[[,<expr>]...]**

Beschreibung

Der ab Adresse <address> spezifizierte Speicherbereich wird bis zum angegebenen Ende fortlaufend mit der Werteliste gefüllt. Die Werte müssen in das Speicherformat passen, ansonsten werden sie modulo <format> berechnet.

Der Debugger führt kein Kontroll-Lesen durch.

Beispiel

Speicherbereich von 1A4:0 bis 1A4:FF mit 55H beschreiben

```
DEBUG_T>FILL BYTE 1A4:0 TO 0FF 55 <CR>
DEBUG_T>
```

C.16 FREETASK

Funktion **Freigeben einer dynamisch geladenen Task**

Syntax **FREETASK <task_id>**

Beschreibung

Mit diesem Kommando wird eine Task gelöscht, die mit dem Kommando `loadtask` oder durch den CLI geladen wurde.

Der Aufruf ist nur erlaubt, wenn sich die mit `<task_id>` angegebene Task im Zustand DORMANT befindet. Das Loader-Result-Segment, sowie alle Speicherbereiche der Task, die mit dem Loader-Result-Segment in Verbindung stehen, werden wieder freigegeben.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

C.17 GO

Funktion **Unterbrochene Task freigeben und Debugger beenden**

Syntax **GO [<start_address>] [$<address1>] [$<address2>]]**

<start_address>	Task-Einsprung-Adresse
<address1>	Soft-Breakpoint-Adresse
<address2>	Soft-Breakpoint-Adresse

Beschreibung

Nur im Breakpoint-Kontext:

GO übergibt die Steuerung der Task, die per Breakpoint unterbrochen wurde. Die Task wird bei <start_address> fortgesetzt, falls angegeben. Fehlt die Adreßangabe, wird die Task ab der Adresse fortgesetzt, die im CS:EIP-Registerpaar gespeichert ist. Mit den optionalen Argumenten $<address1>, $<address2> können ein oder zwei Soft-Breakpoints $0X, $0Y gesetzt werden, bevor die Task gestartet wird. Beide Breakpoints werden automatisch gelöscht, sobald der Debugger wieder die Kontrolle erlangt. Ansonsten verhalten sich beide Breakpoints wie normale Soft-Breakpoints. Das Erreichen dieser Breakpoints wird mit $0X bzw. $0Y angezeigt.

Nach GO im Monitor-Modus wird das unterbrochene Programm, z.B. ein RMOS-Treiber, entsprechend den Registerwerten CS:EIP wieder fortgesetzt. Sofern mit MONITOR in den Monitor-Modus geschaltet wurde, kann mit GO wieder in den Task-Modus gewechselt werden.

Durch Verändern der CS:EIP Registerwerte kann die Programmausführung an einer anderen definierten Einsprungstelle, z.B. zum Neustarten des Systems, wieder fortgesetzt werden.

Beispiel

Die durch den Soft-Breakpoint ($02) angehaltene Task wird mit GO fortgesetzt und gleichzeitig wird an Adresse 0034:0000323A ein neuer Soft-Breakpoint gesetzt.

```
$2, TASK: 0002
DEBUG_T>GO $0034:323A<CR>
```

C.18 HALT

Funktion **Task anhalten**

Syntax **HALT <task_id>**

Beschreibung

HALT hält die Task mit ID <task_id> an und schaltet sie in den Zustand WARTEND. Ferner wird sie mit DEBUGGER ANGEHALTEN (Status 26) gekennzeichnet.

HALT kann nur auf Tasks angewandt werden, die sich Zustand BEREIT befinden.

Hinweis

Wenn ID die eigene Task oder eine nicht anhaltbare Task (wird bei der Erzeugung der Task festgelegt) angibt, wird die Meldung NO SUCH TASK! ausgegeben.

Dieses Kommando ist nur im Task-Modus verfügbar.

Beispiel

Task 3 anhalten

```
DEBUG_T>HALT 3 <CR>
DEBUG_T>
```

C.19 HELP

Funktion **Hilfe-Kommando**

Syntax **HELP [<SVC>|<command>]**

Beschreibung

HELP <SVC> listet die Namen aller Systemaufrufe auf, die mit dem Kommando SVC aufgerufen werden können.

HELP allein listet eine Übersicht aller Kommandos auf.

HELP <command> gibt die Syntax des gewünschten Kommandos wieder. Die Großbuchstaben innerhalb der Kommandonamen zeigen dabei die Anzahl Buchstaben an, mit der ein Kommando eindeutig identifiziert wird. Das dem Kommandonamen folgende YES oder NO gibt an, ob das Kommando konfiguriert ist oder nicht.

Beispiel

Wie lautet die Syntax von HELP ?

```
DEBUG_T>HELP HELP <CR>
SYNTAX: HELP <command> <CR>
DEBUG_T>
```

Welche Kommandos sind konfiguriert?

```
DEBUG_T>HELP <CR>
RMOS3 DYNAMIC DEBUGGER, Vm.n
DEBUG-CMD INC DEBUG-CMD INC DEBUG-CMD INC  ...
---------------------------------------- ...
ASm       YES  BAse     NO   BREaks   YES  ...
...
HElp      YES  ...
...
DEBUG_T>
```

C.20 IN

Funktion **E/A-Port einlesen**

Syntax **IN [{byte|word|dword}] <port_address><port_address> <expr>**

Beschreibung

Vom E/A-Port <port_address> wird im angegebenen Format ein Wert eingelesen. Ohne Formatangabe wird im voreinstellten Byteformat gelesen.

Beispiel

Byte-Wert vom E/A-Port 0EAH einlesen.

```
DEBUG_T>IN BYTE 0EA <CR>
INPUT 00EAH=41H 'A'
DEBUG_T>
```

C.21 INHIB

Funktion **Task-Bearbeitung erlauben/sperren**

Syntax **INHIB {0|1}**

Beschreibung

INHIB erlaubt mit INHIB=0 eine Task-Bearbeitung und sperrt die Bearbeitung mit INHIB=1. Alle Taskzustände bleiben erhalten. SVCs, die von einer Task angefordert wurden, werden beendet, wenn zu deren Ausführung keine weitere Task-Bearbeitung notwendig ist.

Beispiel

```
DEBUG_T>INHIB 1 <CR>
DEBUG_T>
  .
  .
  .
DEBUG_T>INHIB 0 <CR>
DEBUG_T>
```

C.22 KILL

Funktion **Breakpoint(s) löschen**

Syntax **KILL <breakpoint>**

Beschreibung

`KILL` löscht den bezeichneten Breakpoint, d.h. er wird aus der Liste der gesetzten Breakpoints entfernt. Beim Beenden des Debuggers mit `EXITK` werden alle gesetzten Breakpoints gelöscht.

Beispiel

Hard-Breakpoint @3 löschen

```
DEBUG_T>KILL @3 <CR>
DEBUG_T>
```

C.23 LINES

Funktion **Seitenweise Ausgabe ein/ausschalten**

Syntax **LINES <nr>**

Beschreibung

Mit dem Kommando LINES können Sie einstellen, daß die Ausgabe von Befehlen nach jeder vollen Bildschirmseite anhält und mit <Return> fortgesetzt werden kann. Mit <nr> teilen Sie dem Debugger die Anzahl der Zeilen mit, die Ihr Bildschirm aufweist. Die Anzahl muß mindestens 2 betragen und muß dezimal angegeben werden, unabhängig von der Einstellung mit BASE .
LINES 0 schaltet die seitenweise Ausgabe wieder aus.

Beim erstmaligen Start des Debuggers ist die seitenweise Ausgabe ausgeschaltet.

Hinweis

Die Ausgabe des Kommandos REPORT bleibt von der Einstellung der seitenweisen Ausgabe unbeeinflußt.

Ein Abbruch der Ausgabe mit <CTRL> C wirkt sich erst nach dem Fortsetzen der Ausgabe mit <Return> aus.

C.24 LOADTASK

Funktion **Task nachladen**

Syntax **LOADTASK <filename> [<priority> [<pool_id>]] [RUN] [DIS]**

<filename>	Dateiname, Pfadangabe nach RMOS-Konvention
<pool_id>	Speicherpool, aus dem die benötigten Speicherbereiche angefordert werden, Default: -1H (HEAP)
<priority>	Priorität der Task 0 - 255, Default: 80H
RUN	Die Task soll nach dem Laden automatisch gestartet werden
DIS	DISMOUNT-Aufruf zum Abmelden des Datenträgers

Beschreibung

Mit diesem Kommando wird eine Task dynamisch geladen und gegebenenfalls gestartet.

Der Programmcode der Task und zugehörige Datenbereiche werden mit der Funktion stl_load (Relocatibler Tasklader) geladen und entsprechend dem <filename>_<Task-ID>-String (ohne Extension) katalogisiert. Nach Abschluß des Ladevorgangs wird eine dynamische Task kreiert wobei der TCD mit den Registerwerten aus dem Loader-Result-Segment bzw. nach den Parametern in der Kommandozeile vorbesetzt wird.

Die Task-ID und der Selektor für das Loader-Result-Segment sowie die Task-Register und die TCD-Adresse werden auf der Konsole ausgegeben. Sofern der Schalter RUN angegeben wurde, wird die Task nach dem Laden gestartet. Das Starten der Task kann jedoch auch separat über START erfolgen. Der Schalter DIS zum Abmelden des Datenträgers muß angegeben werden, wenn der Datenträger z.B. eine Diskette, zwischenzeitlich gewechselt wird. (Damit ist sichergestellt, daß vor dem Laden automatisch ein neuer MOUNT-Aufruf durchgeführt wird.)

Wenn sich eine nachgeladene Task selbst löscht, (SVC RmDeleteTask), müssen mit FREETASK die von der Task belegten Speicherbereiche trotzdem wieder freigegeben werden.

An ein mit LOADTASK geladenes Programm können mit dem Kommando START Parameter in der Kommandozeile übergeben werden.

Beispiel

```
DEBUG_T>LOADTASK c:rmostask.EXE 0A0 CR
Task-id = 12, Loader result segment = 0180:0
Task entry point = 008C:000099A4,
TCD address = 0030:00000160
DS = 0084, ES = 0084, stack base = 0074:00001000
```

```
DEBUG_T>DIR <CR>(

Cataloged resources

Symbolic-name   kind    id        Symbolic-name   kind    id
DEB_0           TASK    0000H     RMOSTASK_0012   TASK    0012H
DEBUG_T>
```

C.25 MONITOR

Funktion **Debugger in den Monitor-Modus schalten**

Syntax **MONITOR**

Beschreibung

Der Debugger wird, falls er sich im Task-Modus befindet, in den Monitor-Modus geschaltet. Der Prompt wechselt vom Task-Modus-Prompt

```
DEBUG_T> bzw. DEBUG_TB>
```

in den Monitor-Modus-Prompt.

```
DEBUG_MB>
```

Befindet sich der Debugger bereits im Monitor-Modus wird eine Fehlermeldung ausgegeben.

C.26 OUT

Funktion **E/A-Port ausgeben**

Syntax **OUT [{byte|word|dword}] <port_address> <expr>**
<port_address> <expr>

Beschreibung

Auf dem E/A-Port <port_address> wird das mit <expr> angegebene Datenelement im spezifizierten Format ausgegeben (Default: Byteformat).

Beispiel

Wert 1234H auf den Port 5678H ausgeben.

```
DEBUG_T>OUT WORD 5678, 1234<CR>
DEBUG_T>
```

C.27 QUALIFY

Funktion **Breakpoint parametrieren**

Syntax **QUALIFY <breakpoint>,<count>[<task_id>] [{MON|TASK}]**

<breakpoint>	{$|@|E|{M|W}{B|W|D}}{1...0F|40...07F} die Breakpoint-IDs 40..7FH werden von Organon XDB verwendet
<count>	<expr>, Wert für den Durchlaufzähler
<task_id>	ID der Task, für die der Breakpoint gilt
MON	nach Erreichen des Breakpoints in den Monitor-Modus schalten
TASK	nach Erreichen des Breakpoints in den Task-Modus schalten

Beschreibung

QUALIFY setzt den Durchlaufzähler für einen Breakpoint, d.h. der Breakpoint wird erst nach <count> Durchläufen ausgeführt und zur Anzeige gebracht. Mit der optionalen Angabe <task_id> kann die Gültigkeit des Breakpoints auf eine einzelne Task beschränkt werden.

Bei einer Task-ID ANY (= 0FFFFH, Default) gilt der Breakpoint für alle Tasks. Taskspezifische Breakpoints werden verwendet, wenn ein gemeinsamer Codebereich von mehreren Tasks benutzt wird.

Beispiel

Durchlaufzähler von Breakpoint @4 wird auf 7 gesetzt. Der Breakpoint gilt nur für Task 3.

```
DEBUG_T>QUALIFY @1 7 3<CR>

DEBUG_T>BREAKS <CR>

@ 01:  Addr: 2000:0002300F pass count: 0007,task: 003, taskmode
@ 02:  Addr: 2000:00000002 pass count: 0001,task: any, taskmode
E 03:  Addr: 2000:0002345F pass count: 0001,task: any, taskmode
MB04:  Addr: 2000:00023450 pass count: 0001,task: any, taskmode
MW05:  Addr: 2000:00023450 pass count: 0001,task: any, taskmode
MD06:  Addr: 2000:00023450 pass count: 0001,task: any, taskmode
WB07:  Addr: 2000:00023450 pass count: 0001,task: any, mon.mode
WW08:  Addr: 2000:00023450 pass count: 0001,task: any, mon.mode
WD40:  Addr: 2000:00023450 pass count: 0001,task: any, taskmode
E 41:  Addr: 2000:0002345F pass count: 0001,task: any, mon.mode

DEBUG_T>
```

C.28 QUERY

Funktion **Task-Status abfragen**

Syntax **QUERY <task_id>**

Beschreibung

QUERY gibt den Status der Task mit ID <task_id>, zusammen mit ihrer Priorität, auf der Konsole aus.

Die möglichen Task-Zustände sind TASK DORMANT, TASK READY, TASK BLOKKED oder TASK ACTIVE. Für eine Task im Zustand BLOCKED wird außerdem der Grund dafür angegeben. Wenn eine Task noch nie ACTIVE war, wird die Priorität und der Task-Zustand mit dem Wert 00H angezeigt.

Beispiel

Status der Task mit ID=2 abfragen

```
DEBUG_T>QUERY 2 <CR>
Task dormant
State byte: 00, current priority: FC.
DEBUG_T>
```

In Bit 0 und 1 des Zustands-Bytes (state byte) ist der aktuelle Zustand einer Task binär codiert:

0 - DORMANT (Ruhend)
1 - READY (Bereit)
2 - BLOCKED (Wartend)
3 - ACTIVE (Rechnend)

Tabelle C-1 Status einer Task im Zustand WARTEND

06H	Auf Ereignisflag wartend
0AH	Auf Semaphore oder auf Zugang zu überwachtem Programm wartend
0EH	Warten bis Zieltask geladen ist
12H	Auf das Starten der Zieltask wartend
16H	Auf das Beenden der Zieltask wartend
1AH	Auf das Empfangen einer Botschaft wartend
1EH	Auf das Empfangen einer gesendeten Botschaft wartend
22H	Auf Zuweisung von Speicherplatz aus einem Speicherpool wartend
26H	Vom Debugger angehalten oder Task im Zustand suspendiert
2AH	Vom Debugger-Breakpoint unterbrochen
2EH	Auf den Ablauf eines Zeitintervalls wartend (PAUSE)
32H	Fehler: Ungültiger Task-Stackselektor

Tabelle C-1 Status einer Task im Zustand WARTEND

36H	Auf das Beenden einer E/A-Operation wartend
3EH	Warten auf Speicherfreigabe (FREEALL)
42H	Warten bis Zeitintervall abgelaufen
46H	Laufzeit-Fehler, Typ 0 (Division by 0 Interrupt)
4AH	Laufzeit-Fehler, Typ 1 (Single Step Interrupt)
4EH	Laufzeit-Fehler, Typ 3 (Breakpoint Interrupt)
52H	Laufzeit-Fehler, Typ 4 (Overflow Interrupt)
56H	Laufzeit-Fehler, Typ 5 (Array Bound Interrupt)
5AH	Laufzeit-Fehler, Typ 6 (Unused Opcode)
5EH	Laufzeit-Fehler, Typ 7 (Escape Opcode)
62H	Laufzeit-Fehler, Typ 8 (Double Fault)
66H	Laufzeit-Fehler, Typ 9 (NDP Segment Overrun)
6AH	Laufzeit-Fehler, Typ 13 (General Protection)
6EH	Laufzeit-Fehler, Typ 16 (Floating Point Error)
72H	Laufzeit-Fehler, Typ 10 (Invalid TSS)
76H	Laufzeit-Fehler, Typ 11 (Segment Not Present)
7AH	Laufzeit-Fehler, Typ 12 (Stack Fault)
7EH	Laufzeit-Fehler, Typ 14 (Page Fault)
82H	Laufzeit-Fehler, Typ 17 (Alignment Check)
C6H	Warten auf das Eintragen eines Catalog-Strings
CAH	Task wird mit KILLTSK beendet (nach Abschluß einer E/A-Operation)
CEH	Task wird mit KILLTSK gelöscht (nach Abschluß einer E/A-Operation)

C.29 REGS

Funktion **Alle Registerinhalte anzeigen**

Syntax **REGS [<task_id>]**

Beschreibung

Im **Breakpoint-Kontext:**

REGS zeigt die Inhalte der im Real Mode verwendeten Register der Prozessoren 8086/80186/80286/80386/80486/Pentium, bzw. die im Protected Mode verwendeten Register der Prozessoren an.

Im **Escape-Kontext:**

REGS zeigt nach Angabe einer Task-ID die Register der Task an (nur sinnvoll, wenn die Task-ID zu einer im Zustand WARTEND befindlichen Task gehört).

Ausgabe der Register im Real Mode:

```
AX: xxxx  CS: xxxx  IP: xxxx
BX: xxxx  SS: xxxx  SP: xxxx  BP: xxxx
CX: xxxx  DS: xxxx  SI: xxxx
DX: xxxx  ES: xxxx  DI: xxxx
FL: xxxx
```

Ausgabe des 80386 im Protected Mode:

```
 CS = ssss  ppppppppL  Limit: llllllll  ER N G  EIP = xxxxxxxx
 SS = ssss  ppppppppL  Limit: llllllll  RW N G  ESP = xxxxxxxx
 DS = ssss  ppppppppL  Limit: llllllll  RW N G  EAX = xxxxxxxx
 ES = ssss  ppppppppL  Limit: llllllll  RW N G  EBX = xxxxxxxx
 FS = ssss  ppppppppL  Limit: llllllll  RW N G  ECX = xxxxxxxx
 GS = ssss  ppppppppL  Limit: llllllll          EDX = xxxxxxxx
 TR = ssss  ppppppppL  Limit: llllllll          EDI = xxxxxxxx
LDT = ssss  ppppppppL  Limit: llllllll          ESI = xxxxxxxx
GDT = ssss  ppppppppL  Limit: llllllll          EBP = xxxxxxxx
IDT = ssss  ppppppppL  Limit: llllllll          EFL = xxxxxxxx
                                                CR0 = xxxxxxxx
```

ssss	Selektor
ppppppppL	lineare (physikalische) 32-Bit-Adresse
llllllll	Segment-Limit
xxxxxxxx	Allgemeiner Registerinhalt

Alle Ausgaben werden hexadezimal dargestellt.

Beispiel

Ein Soft-Breakpoint $2 im Codebereich der Task mit ID=2 wird bei 9C:6C gesetzt. Die Task wird gestartet und danach durch den Soft-Breakpoint $2 angehalten. Anschließend werden die Register angezeigt.

```
DEBUG_T>tcd 2
Task entry point = 009C:0000006C, TCD address = 0038:00091168,
DS = 0000, ES = 0000, stack base = 00E0:00001000
DEBUG_T>set $2 9c:6c
DEBUG_T>start 2 0 0
DEBUG_T>
Breakpoint $02 reached in task 'ERRLOG' (02H).
009C:0000006C   1E                 PUSH   DS
DEBUG_TB>regs

CS = 009C  Linear: 00083580  Limit: 00000110  ER N    EIP = 0000006C
SS = 00E0  Linear: FFC05000 Limit: 00000FFF  RW N    ESP = 00000FF0
DS = 0000  (invalid)                                 EAX = 00000000
ES = 0000  (invalid)                                 EBX = 00000000
FS = 00E0  Linear: FFC05000 Limit: 00000FFF  RW N    ECX = 00000000
GS = 0000  (invalid)                                 EDX = 00000000
TR = 0018  Linear: 0001A3C0 Limit: 00000067          EDI = 002E0A02
LDT = 0020 Linear: 0001A200 Limit: 000000D7          ESI = 00000000
GDT =      Linear: 0000A000 Limit: 0000F9FF          EBP = 0000003A
IDT =      Linear: 00019A00   Limit: 000007FF        EFL = 00243216
                                                     CR0 = 8000001B

DEBUG_TB>go
End of task 'ERRLOG' (02H) detected.
DEBUG_T>
```

Hinweis

Nach dem Start der Task 2 wartet der Debugger nach der Prompt-Ausgabe auf Eingabe von der Konsole. Er wird in diesem Zustand vom Konsolentreiber blockiert und kann daher noch nicht den von der Task 2 erreichten Breakpoint melden. Eine beliebige Eingabe (z. B. Leerzeichen) sorgt für die Meldung inzwischen erreichter Breakpoints sowie für den Wechsel vom Escape- in den Breakpoint-Kontext.

C.30 REPORT

Funktion **Resource Report**

Syntax **REPORT [<resource> [<resource-id>]] [LONG]**

<resource>	[{TASK|SEMA|FLAG|LMBX|MSGQ}]
TASK	Task-Report
SEMA	Ausgabe des Semaphor-Report
FLAG	Report über die Ereignis-Flags
LMBX	Mailbox-Report
MSGQ	Message Queue
<resource-id>	Kennung des Betriebsmittels
[LONG]	Langfassung, detaillierte Ausgabe

Beschreibung

REPORT ruft die Resource Reporter-Funktionen auf. Der Resource Reporter gibt Auskunft über die von RMOS verwalteten Betriebsmittel. Standardmäßig ist eine Kurzfassung der Ausgabe eingestellt, mit dem [Long]-Schalter kann die Langfassung aktiviert werden. Bei fehlender Typ-Angabe werden alle vorhandenen Ressourcen ausgewertet. Die Reporter-Ausgabe kann mit der Angabe von <resource-id> auf ein einziges Betriebsmittel eingeschränkt werden. Die Auswertung erfolgt nicht durch eine eigenständige Resource Reporter-Task sondern im Debugger-Kontext. Die Auswertung erfolgt einmal, ohne zyklische Wiederholung. Die Ausgaben erscheinen auf dem für den Debugger vereinbarten Device-ID und Unit-ID, die für den Resource Reporter bei der Konfigurierung getroffene Vereinbarung wird ignoriert.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

Die Ausgabe von REPORT bleibt von der Einstellung der seitenweisen Ausgabe mit dem Kommando LINES unbeeinflußt.

Vorsicht

Der Resource Reporter ist nicht reentrant, d.h. er darf nicht gleichzeitig in mehreren Debuggern aufgerufen werden, da ansonsten ein Debuggerabsturz auftreten kann.

Beispiel

Kommando zum Listen der Tasks.

Es werden alle Tasks aufgelistet, die sich in den Zuständen READY, ACTIVE oder BLOCKED befinden.

```
DEBUG_T>rep task
RMOS3 RESOURCE REPORTER,  V3.8                 01-FEB-1996 07:28:44
TASK REPORT:
TASK_ID STATE CUR_PRI INH_PRI   TASK_ID    STATE   CUR_PRI   INH_PRI
     18 ready      1       1        1E   i/o oper      40        40
     21  active    FF      FF
DEBUG_T>rep task lo
RMOS3 RESOURCE REPORTER,  V3.8                 01-FEB-1996 07:29:44
TASK REPORT:
TASK_ID 18  STATE ready       ATTR  dyn
CUR_PRI  1  EAX   0  DS    0  START   84:000002DC
INH_PRI  1  EBX   0  ES    0  STACK  208:000003AA  TCD  38:000026A4
TASK_ID 1E  STATE i/o oper    ATTR  dyn
CUR_PRI 40  EAX    0  DS  240  START   44:00004820
INH_PRI 40  EBX    0  ES  240  STACK  240:00001AB4  TCD  38:000027B4
TASK_ID 21  STATE active      ATTR  nhlt dyn
CUR_PRI FF  EAX    0  DS  270  START   94:00003D24
INH_PRI FF  EBX    6  ES   0  STACK  268:00000ECC  TCD  38:00002908
```

Bedeutung

Die Ausgaben haben folgende Bedeutung:

TASK_ID:	ID der Task
STATE:	Zustand der Task
CUR_PRI:	aktuelle Priorität
INH_PRI:	Priorität zum Startzeitpunkt
ATTR:	Attribute der Task
START:	Startadresse der Task
STACK:	aktueller Zeiger auf den momentanen Stack
TCD:	Adresse des TCD
EAX,EBX,DS,ES:	Startwert der jeweiligen Register

Beispiel

Kommando zum Listen der Semaphore.

```
DEBUG_T>rep sema
RMOS3 RESOURCE REPORTER,  V3.8                01-FEB-1996 07:31:20
SEMAPHORE REPORT:
SEMA_ID  IN_USE  TASK_WAIT            SEMA_ID  IN_USE  TASK_WAIT
      0     no         no                   1     no         no
      2     no         no                   3     yes        yes
      4     no         no                   5     no         no
      6     no         no                   7     no         no
      8     no         no                   9     no         no
      A     no         no                   B     no         no
      C     no         no                   D     no         no
DEBUG_T>rep sema lo
RMOS3 RESOURCE REPORTER,  V3.8                01-FEB-1996 07:31:23
SEMAPHORE REPORT:
SEMA_ID  OWNER_TASK   REQ_TASK   REQ_PRI
      0
      1
      2
      3          21         26        FF
                            29        FF
      4
     ...
      C
      D
```

Bedeutung

Die Ausgaben haben folgende Bedeutung:

SEMA_ID:	ID der Semaphore
IN_USE:	Semaphore belegt? (yes/no)
TASK_WAIT:	Warten Tasks auf die Zuteilung der Semaphore? (yes/no)
OWNER_TASK:	Task ID, welche die Semaphore belegt hat
REQ_TASK:	Task ID von wartenden Tasks
REQ_PRI:	Priorität von wartenden Tasks

Beispiel

Kommando zum Listen der Flaggruppen.

```
DEBUG_T>rep flag
RMOS3 RESOURCE REPORTER,  V3.8                 01-FEB-1996 07:31:56
EVENT_FLAG REPORT:
EVENT_FLG  OWNER   FLAGS  TASK_WAIT   EVENT_FLG  OWNER    FLAGS
TASK_WAIT
    local     18       0        no      local     1E      0       no
    local     21       1        no          1             0       no
        2             0        no          3             0       no
        4             0        no          5             0       no
        6             0        no          7             0       no
        8             0        no          9             0       no
        A             0        no
DEBUG_T>rep flag lo
RMOS3 RESOURCE REPORTER,  V3.8                 01-FEB-1996 07:31:59
EVENT_FLAG REPORT:
EVENT_FLG  OWNER    FLAGS  TASK_WAIT  PRI  TEST_MASK  TYPE
    local     18        0
    local     1E        0
    local     21        1
        1             0
        2             0
        ...           0
        9             0
        A             0
```

Bedeutung

Die Ausgaben haben folgende Bedeutung:

EVENT_FLG:	ID der Eventflag Gruppe oder ”local” für lokale Eventflags
OWNER:	Eigentümer der lokalen Eventflaggruppen
FLAGS:	Zustand der Flags
TASK_WAIT:	Warten Tasks auf die Zuteilung der Semaphore? (yes/no)
PRI:	Priorität von wartenden Tasks
TEST_MASK:	Flagmaske der Wartenden Tasks
TYPE:	Verknüpfungsart der Testmaske mit der Flaggruppe (and/or)

Beispiel

Kommando zum Listen der Mailboxen.

```
DEBUG_T>rep lmbx
RMOS3 RESOURCE REPORTER,  V3.8                01-FEB-1996 07:32:23
LOCAL MAILBOX REPORT:
MBOX_ID  TASK_WAIT  MSG_AVAIL         MBOX_ID  TASK_WAIT  MSG_AVAIL
      0        no         no               1        no         no
      2        no         no               3        no         no
      4        no         no               5        no         no
      6        no         no               7        no         no
      8        no         no               9        no         no
DEBUG_T>rep lmbx lo
RMOS3 RESOURCE REPORTER,  V3.8                01-FEB-1996 07:32:27
LOCAL MAILBOX REPORT:
MBOX_ID  RECVR  SENDER  PRI     COORD      MSG1     MSG2     MSG3
      0
      1
      2
      ....
```

Bedeutung

Die Ausgaben haben folgende Bedeutung:

MBOX_ID:	ID der Mailbox
TASK_WAIT:	Warten Tasks auf die Zuteilung der Semaphore? (yes/no)
MSG_AVAIL:	Sind Nachrichten in der Mailbox vorhanden? (yes/no)
RECVR:	Empfängertask, die auf Nachrichten wartet
SENDER:	Sendertask, die auf Abholen der Nachricht wartet
PRI:	Priorität von wartenden Tasks
COORD:	nicht relevant
MSG1,MSG2,MSG3:	Inhalt der Nachricht (3 32Bit Worte)

Beispiel

Kommando zum Listen der Message-Queues.

```
DEBUG_T>rep msgq

RMOS3 RESOURCE REPORTER,  V4.0                17-MAR-1998 11:16:04
MESSAGE QUEUE REPORT:
TASK_ID  TASK_WAIT  MSG_AVAIL       TASK_ID  TASK_WAIT  MSG_AVAIL
      3        no      5                 22        no      0
     23        yes     0                 28        yes     0
     2A        yes     0
```

Beispiel für REP MSGQ LONG

```
DEBUG_T>rep msgq long

RMOS3 RESOURCE REPORTER,  V4.0                 17-MAR-1998 11:16:06
MESSAGE QUEUE REPORT:
TASK_ID  SENDER  PRI     COORD      MESSAGE_ID    MESSAGE
      3      2C    3   no coord              3    0000:00000003
             2C    2   no coord              2    0000:00000002
             2C    1   no coord              1    0000:00000001
             2C    0   no coord              0    0000:00000000
             23    0   no coord             64    0000:00005555
     22
     23
     28
     2A
```

Bedeutung

Die Ausgaben haben folgende Bedeutung:

TASK_ID = Task-ID der Task, der die Message-Queue gehört.

TASK_WAIT = Wartet die Task auf eine Message? (yes/no)

MSG_AVAIL = Anzahl der Messages in der Queue

SENDER = TASK-ID der Task, die die Message gesendet hat

PRI = Priorität der Message

COORD = (siehe REP LMBX LONG)

MESSAGE_ID = ID der Message (bei RmSendMessage Parameter Message)

MESSAGE = Message (bei RmSendMessage Parameter pMessageParam)

Hinweis

Der Selector (Wert links vom ':') wird bei Flat-Programmen ignoriert, bzw. kann mit RmSendMessage nicht gesetzt werden.

C.31 SET

Funktion

Breakpoint(s) setzen

Syntax

SET <breakpoint>,<address>

<breakpoint> {$|@|E|{M|W}{B|W|D}}1...0F

$	Soft-Breakpoint setzen
@	Hard-Breakpoint setzen

Breakpoint im Debugregister setzen:

E	Execution-Breakpoint im Debugregister
MB	Modify (Read or Write) Byte Breakpoint
MW	Modify (Read or Write) Word Breakpoint
MD	Modify (Read or Write) Dword Breakpoint
WB	Write Byte Breakpoint
WW	Write Word Breakpoint
WD	Write Dword Breakpoint

<address> Selektor:Offset oder lineare Adresse

Beschreibung

SET wird verwendet, um Breakpoints unter Verwendung der INT 3-Anweisung bzw. der Debugregister des Prozessors zu setzen.

SET {$|@} ersetzt das Byte bei <address> durch eine INT 3-Anweisung. Der entsprechende Programmcode muß sich im RAM-Bereich befinden. Soft-Breakpoints ($) sind nur temporär vorhanden d.h. sie werden automatisch nach der Abarbeitung gelöscht. Im Gegensatz dazu bleiben Hard-Breakpoints (@) erhalten, bis sie mit KILL entfernt werden. Wenn nach Erreichen eines Hard-Breakpoints die Programmausführung wieder fortgesetzt wird, ersetzt der Debugger die INT 3-Anweisung durch den ursprünglichen Opcode. Nach Abarbeitung dieses Befehls im Einzelschritt-Modus wird wieder die INT 3-Anweisung eingetragen. Die Task-Ausführung wird anschließend wieder mit normaler Geschwindigkeit fortgesetzt.

SET {E|{{M|W}{B|W|D}} setzt einen Breakpoint unter Zuhilfenahme der Debugregister des Prozessors. Es können maximal 4 Debugregister-Breakpoints gesetzt werden. Mit diesem Breakpointtyp können sowohl Execution-Breakpoints im EPROM-Bereich gesetzt als auch Breakpoints für Schreib-/Lese-Speicherzugriffe definiert werden. Ein mit SET definierter Breakpoint ist für alle Tasks gültig und wird nach erstmaligem Erreichen ausgeführt. (Siehe QUALIFY für Hinweise zum Setzen des Durchlaufzählers und der Task-Einschränkung.) Beim Erreichen eines Breakpoints übernimmt der Debugger die Steuerung im **Breakpoint-Kontext** und gibt eine entsprechende Meldung an der Konsole aus.

Die Vergabe von Breakpoint-Nummern bei SET ist frei wählbar, d.h. es kann z.B. Breakpoint-Nummer 8 vor Breakpoint-Nummer 5 verwendet werden.

Beispiel

INT3-Breakpoint:
Soft-Breakpoint an Adresse 2000:0 ($01),
Hard-Breakpoints an Adressen 11B5:2 und 2000:0F (@02 und @03), setzen.

```
DEBUG_T>SET $01,2000:0 <CR>
DEBUG_T>SET @02,11B5:2 <CR>
DEBUG_T>SET @03,2000:0F <CR>
DEBUG_T>
```

Debugregister-Breakpoint:
Execution-Breakpoint auf Adresse 01A4:0000E49A (E05)
Modify (Read or Write) auf Adresse 01A4:000032F5 (MB06)
Modify (Write) auf Adresse 01A4:00012A24 (WW07)

```
DEBUG_T>SET E05, 01A4:0000E49A <CR>
DEBUG_T>SET MB06,01A4:000032F5 <CR>
DEBUG_T>SET WW07,01A4:00012A24 <CR>
```

C.32 STACK

Funktion **Stack anzeigen**

Syntax **STACK [<count>]**

Beschreibung

Nur im Breakpoint-Kontext:

STACK gibt ab Adresse SS:ESP <count> Stackeinträge aus.

In einem 32-Bit-Segment werden die Stackeinträge als DWORDs, in einem 16-Bit-Segment als WORDs ausgegeben.

Beispiel

```
DEBUG_T>STACK 10<CR>

SS:      0074     Linear:  00174BC  Limit:   00000322 ESP:   00001E0
FFFF006C 00000000 00000000 FFFF006C 000002D0 0000000C 00000308 00000210
00000000 00001456 000023EA FFFF0000 00000383 0000FFFF 0000F0F0 0000AAAA
```

C.33 START

Funktion **Task starten, die im Zustand DORMANT ist**

Syntax **START <task_id>[{<address>|<expr>[<expr>] | ARGS [<arg> ...]}]**

Beschreibung

START schaltet eine Task mit ID <task_id> im Zustand DORMANT in den Zustand BEREIT, wobei die Werte in <address> bzw. in <expr>[<expr>] als Startparameter benutzt werden. Bei Angabe einer <address> als Startparameter wird der Offset ins EAX-Register und das Segment ins EBX-Register übertragen. Bei Angabe von <expr> [<expr>] als Startparameter wird der erste Wert ins EAX-Register und der zweite Wert ins EBX-Register geschrieben. Werden die Startparameter weggelassen, übernimmt der Debugger die beim letzten START angegebenen Parameter (ursprünglich Null). Wird nur der erste Parameter angegeben (EAX-Register), wird für den zweiten Parameter der zuletzt gültige Wert verwendet.

Wird die zweite Form mit dem Schlüsselwort ARGS verwendet, so werden alle weiteren Argumente in der Form aufbereitet, daß sie der gestarteten Task als ”argc/argv” zur Verfügung stehen. argv[0] enthält den Dateinamen, der bei LOADTASK angegeben wurde oder, falls das Programm nicht mit LOADTASK geladen wurde, den Katalogeintrag oder, wenn keiner existiert, den String ”(UNKNOWN)”. Die Kommandozeile sowie argv und argc werden auf dem Stack der Task abgelegt. Der Stack muß dementsprechend viel Platz aufweisen.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

Beispiel

Die Errorlogger-Task besitzt die Task-ID 6. Ab Adresse 002C:0 soll ein freier RAM-Bereich liegen. Das folgende Beispiel trägt ins RAM den Textstring ABCD ein, der anschließend von der Errorlogger-Task an die Konsole ausgegeben wird.

```
DEBUG_T>CHANGE BYTE 002C:0 41,42,43,44<CR>
DEBUG_T>START 6 002C:0 <CR>
DEBUG_T>EXIT
ABCD
```

C.34 STEP

Funktion **Einzelschritt ausführen**

Syntax **STEP [<count>]**

`<count>` Wiederholungszähler, maximal 127

Beschreibung

Nur im Breakpoint-Kontext:

Die unterbrochene Task wird im Einzelschritt-Modus ausgeführt.

Soll mehr als ein Prozessorbefehl mit STEP ausgeführt werden, kann mit dem Parameter `<count>` die Anzahl der Befehle angegeben werden, die pro STEP durchlaufen werden.

Wenn während der Einzelschritt-Bearbeitung ein Interrupt oder eine Exception auftritt, wird die Interrupt-Bearbeitung ohne Unterbrechung ausgeführt. Um eine Interrupt-Routine im Einzelschritt testen zu können, muß in dem entsprechenden Codebereich ein Breakpoint eingetragen werden.

Es ist nicht möglich, mit STEP in einen SVC hineinzulaufen. SVC-Aufrufe werden wie ein einziger Assembler-Befehl abgearbeitet. Die Beendigung eines STEPs kann verzögert werden, z.B. durch einen blockierenden SVC oder durch den Ablauf höherpriorer Tasks.

C.35 SVC

Funktion **Systemaufruf generieren**

Syntax **SVC <SVC_name> <param_list> oder kurz:**
<SVC_name> <param_list>

`<SVC_name>`	gültiger SVC-Name (siehe Kommando `HELP SVC`)
`<param_list>`	Liste der SVC-Parameter

Beschreibung

Der Debugger generiert einen SVC an M7 RMOS32. Die Parameterliste besteht aus dem ausgeschriebenen SVC-Namen (nur soweit bis eindeutig) und zugehörigen Argumenten (als Bytes, Worte oder Adressen).

Alle Argumente werden entsprechend der SVC-Hochsprachen-Schnittstelle übergeben, d.h. die Parameter müssen wie bei einem SVC in der Hochsprache C in die Kommandozeile eingegeben werden. Anzahl und Inhalt der Parameter werden überprüft, fehlerhafte Eingaben werden zur Anzeige gebracht und das Kommando wird abgebrochen. Bei korrekter Parameter-Eingabe wird der SVC ausgeführt und nach Beendigung eine Meldung ausgegeben, ob der Aufruf erfolgreich ausgeführt worden konnte, gefolgt vom Rückgabewert.

Erwartet ein SVC die Eingabe eines Strings, kann statt eines Pointers der String direkt eingegeben werden. Erwartet ein SVC die Eingabe eines Pointers, um dort einen Rückgabewert abzulegen, so kann bei einfachen Datentypen statt des Pointers das Schlüsselwort AUTO eingegeben werden (bzw. bei interaktiver Eingabe einfach leer Return gedrückt werden). Dann stellt der Debugger selbständig Platz für das Datum zur Verfügung und übergibt einen Zeiger darauf an den SVC. Hinterher, bei Rückgabe von Pointern, Zahlen, Strings, Mails. Doppelwörtern, MempoolInfoStruct, TaskInfoStruct, gibt der Debugger den zurückgelieferten Wert aus.

Hinweis

Beim Aufruf `RmCreateDescriptor` muß die Adresse als long-Wert angegeben werden (z.B. 28000). Dieser long-Wert gibt die absolute physikalische Segment-Basisadresse an. Der Adreßzusatz l oder p (28000l/28000p) ist nicht zulässig.

Dieses Kommando ist nur im Task-Modus verfügbar.

API-Aufrufe, die nicht mit SVC abgesetzt werden können: `RmReadMessage`

Beispiel

```
>svc rmsetflag 0 14
SVC successful.
SVC status -259


svc rmalloc 0 0 300000 0e0:0
SVC failed:
Error code 38 = 'Invalid size'
```

oder kurz:

```
>rmsetflag 0 14
SVC successful.
SVC status -259
```

C.36 SWITCH

Funktion **In Breakpointkontext wechseln**

Syntax **SWITCH <task_id>**

Beschreibung

Sind mehrere Tasks an Breakpoints angehalten worden, kann mit diesem Kommando aus dem Escape-Kontext oder dem aktuellen Breakpointkontext einer Task in den Breakpointkontext der mit `task_id` gekennzeichneten Task gewechselt werden. Kommandos wie z.B. REGS, GO, STEP beziehen sich nun auf diese Task.

C.37 TASK

Funktion **Debugger in den Task-Modus schalten**

Syntax **TASK**

Beschreibung

Im **Escape-Kontext:**

Der Debugger wird, falls er sich im Monitor-Modus befindet, in den Task-Modus geschaltet. Der Prompt wechselt vom Monitor-Modus-Prompt
`DEBUG_MB>`
in den Task-Modus-Prompt.
`DEBUG_T>`

Befindet sich der Debugger bereits im Task-Modus, wird eine Fehlermeldung ausgegeben.

Im **Breakpoint-Kontext**:

Eine direkte Umschaltung in den Task-Modus ist nicht möglich. Der Monitor-Modus kann nur mit dem Go-/Exit-Kommando verlassen werden. Damit wird gleichzeitig das unterbrochene Programm fortgesetzt.

C.38 TCB

Funktion **Daten vom TCB einer Task anzeigen**

Syntax **TCB <task_id>**

Beschreibung

TCB gibt die aktuellen, dynamisch veränderbaren Daten des Task-Steuerblocks (TCB) der Task mit ID `<task_id>` an der Konsole aus. Die Daten im TCB werden für statische Tasks bei der Systemkonfigurierung und bei dynamisch kreierten Tasks zur Laufzeit des Systems definiert.

Mit diesem Kommando kann u.A. das LOADER-Segment abgefragt werden, das benötigt wird, um eine bereits laufende Task mit dem Organon XDB zu debuggen.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

C.39 TCD

Funktion **Daten vom TCD einer Task anzeigen**

Syntax **TCD <task_id>**

Beschreibung

TCD gibt einige Daten des Task-Steuerblocks (TCD) der Task mit ID `<task_id>` an der Konsole aus. Die Daten im TCD werden für statische Tasks bei der Systemkonfigurierung und bei dynamisch kreierten Tasks zur Laufzeit des Systems definiert.

Hinweis

Dieses Kommando ist nur im Task-Modus verfügbar.

Beispiel

Protected Mode: TCD der Task mit ID=2 anzeigen

```
DEBUG_T>TCD 2 <CR>
Task entry point = 11B5:0002, TCD Address = 1004:0040,
DS = 0000, ES = 0000, stack base = 23FB:00D0
DEBUG_T>
```

```
DEBUG_T>TCD 2 <CR>
Task entry point = 005C:000024F4, TCD Address = 0014:00000362,
DS = 0000, ES = 0000, stack base = 0064:00000320
DEBUG_T>
```

C.40 Fehlermeldungen des Debuggers

Der Debugger antwortet auf fehlerhafte Eingaben mit einer Fehlermeldung:

Fehlermeldung	Bedeutung
']' expected	In einem Ausdruck fehlt eine rechte eckige Klammer.
All debugregisters in use!	Es wurde versucht, mehr als 4 Breakpoints vom Typ 'Debugregister Breakpoint' zu setzen.
Already a breakpoint!	Die Adresse ist bereits von einem Breakpoint besetzt.
Balance error!	Ungleiche Anzahl öffnender und schließender Klammern.
Breakpoint_id in use!	Die Breakpoint-Nummer (ID) kann nicht überschrieben werden.
Can't allocate memory!	Es konnte kein oder nicht ausreichend Speicher allociert werden.
Can't monitor end of task!	Der Debugger konnte sich beim Betriebssystem nicht für die Endeüberwachung dieser Task anmelden.
Catalog error!	Der Debugger konnte einen Katalogeintrag nicht erzeugen.
Command not allowed on this unit!	Es wurde versucht in den Monitor-Modus zu schalten oder einen Monitor-Breakpoint an einer Konsole zu setzen, die diese Betriebsart nicht unterstützt.
Command not configured!	Das Kommando wurde nicht konfiguriert.
Divide by zero!	Ein Ausdruck enthält eine Nulldivision.
Error starting task	Eine Task konnte nicht gestartet werden.
I/O- error!	Ein Ein-/Ausgabefehler ist aufgetreten.
Illegal mode!	Es wurde versucht, im Task-Mode ein Kommando auszuführen, das nur im Monitor-Mode erlaubt ist, bzw. es wurde versucht, im Monitor-Mode ein Kommando auszuführen, das nur im Task-Mode zugelassen ist.
Illegal request!	Das Kommando ist im derzeitigen Zustand nicht zugelassen.
Illegal SVC parameter	Ungültige Parametereingabe beim SVC-Kommando.
Invalid base!	Ungültige Zahlenbasis.
Loader error	Unbekannter Fehler des Task-Laders
Loader result segment not found!	Für die angegebene Task existiert kein Loader-Result-Segment.
Memory not writeable	Es wurde versucht, einen Hard- oder Soft-Breakpoint in einem ROM-Speicherbereich zu setzen.
Missing operand(s)!	In einem Ausdruck fehlt ein gültiger Operand.
Missing valid SVC_name!	Der SVC ist entweder fehlerhaft oder existiert nicht.
Nesting exceeded, stack full!	In einem Integer- oder Gleitpunkt-Ausdruck ist die Schachtelungstiefe zu groß.
No linear/physical address for breakpoints!	Beim Setzen eines Breakpoints ist eine lineare/physikalische Adresse nicht erlaubt.
No such breakpoint!	Es wurde versucht, einen Breakpoint zu löschen, der nicht existiert.
No such task!	Die Task-ID wurde keiner Task zugewiesen oder die Task befindet sich nicht im Zustand DORMANT.
Not a read-write segment!	Das mit einem Selektor angegebene Segment muß vom Typ READ-WRITE sein.

`Not an execute-read segment!`	Das mit einem Selektor angegebene Segment muß vom Typ EXECUTE-READ sein.
`NPX not allowed!`	Die Coprozessor-Operation ist nicht zugelassen.
`Offset exceeds segment limit!`	Das Segment-Limit wurde überschritten.
`Page not present!`	Für diesen Speicherbereich existiert kein Eintrag in dem Page-Directory oder die zugehörige Page ist nicht vorhanden.
`Stepcount > 127!`	Mit STEP können maximal 127 Befehle ausgeführt werden
`Syntax error!`	Das zuletzt eingegebene Kommando ist syntaktisch falsch.
`Task not halted!`	• Task ist nicht im Zustand DEBUGGER HALTED oder DEBUGGER BREAKPOINT (Kommando switch oder cont) oder • Task ist nicht im Zustand DORMANT (Kommando freetask oder start) oder • Debugger nicht in Breakpoint-Kontext (Kommando step)
`Too many arguments!`	Bei START <task_id> ARGS ... und SVC <SVC_name> wurden zuviele Argumente angegeben.
`Type mismatch!`	Ungültige TYP-Angabe.
`Wrong report table size!`	Falsche Größe der Resource Reporter-Tabelle.
`Wrong selector!`	Es wurde versucht ein Selektor/Offset-Paar als Adresse anzugeben, dessen Selektor-Teil auf keinen gültigen Eintrag in einer Descriptortabelle zeigt.

Systemzustandsliste SZL

D

Kapitelübersicht

Im Kapitel	finden Sie	auf Seite
D.1	Überblick über die Systemzustandsliste (SZL)	D-2
D.2	Aufbau einer SZL-Teilliste	D-3
D.3	SZL-ID	D-4
D.4	Mögliche SZL-Teillisten	D-5
D.5	SZL-ID W#16#xy11 - Baugruppen-Identifikation	D-6
D.6	SZL-ID W#16#xy12 - CPU-Merkmale	D-7
D.7	SZL-ID W#16#xy13 - Anwenderspeicherbereiche	D-9
D.8	SZL-ID W#16#xy14 - Systembereiche	D-10
D.9	SZL-ID W#16#xy15 - Bausteintypen	D-11
D.10	SZL-ID W#16#xy22 - Alarmstatus	D-13
D.11	SZL-ID W#16#xy24 - Betriebszustand und Betriebszustands-übergang	D-14
D.12	SZL-ID W#16#xy32 - Kommunikationszustandsdaten	D-18
	Die Abschnitte D.13 bis D.15 sind Datensätze aller Auszüge zur Teilliste W#16#0132	
D.16	SZL-ID W#16#xy91 - Baugruppenzustandsinformation	D-22
D.17	SZL-ID W#16#xy92 - Baugruppenträger-/Stationszustandsinfor-mation	D-24
D.18	SZL-ID W#16#xyA0 - Diagnosepuffer	D-26
D.19	SZL-ID W#16#00B1 - Baugruppendiagnoseinfo	D-27
D.20	SZL-ID W#16#00B2 - Baugruppendiagnosedaten über geogra-phische Adresse	D-28
D.21	SZL-ID W#16#00B3 - Baugruppendiagnosedaten über logische Basisadresse	D-29
D.22	SZL-ID W#16#00B4 - Diagnosedaten eines DP-Slaves	D-30

D.1 Überblick über die Systemzustandsliste (SZL)

In diesem Anhang

In diesem Anhang sind alle Teillisten der Systemzustandsliste beschrieben, deren Auskünfte sich auf

- CPUs beziehen oder
- auf solche Baugruppen, deren Teillisten nicht baugruppenspezifisch sind (z. B. SZL-IDs W#16#00B1, W#16#00B2, W#16#00B3).

Baugruppenspezifische Teillisten z. B. für CPs und FMs entnehmen Sie bitte der jeweiligen Baugruppenbeschreibung.

Definition: Systemzustandsliste

Die Systemzustandsliste beschreibt den aktuellen Zustand eines Automatisierungssystems. Der Inhalt der SZL kann durch Auskunftsfunktionen nur gelesen, nicht aber geändert werden. Die Teillisten sind virtuelle Listen, das heißt, sie werden vom Betriebssystem der Zentralbaugruppen nur auf Anforderung zusammengestellt.

Inhalt

Die Systemzustandsliste enthält Informationen über:

- Systemdaten
- Diagnosezustandsdaten in der CPU
- Diagnosedaten auf Baugruppen
- Diagnosepuffer

Systemdaten

Systemdaten sind feste oder parametrierte Kenndaten einer CPU.

Sie beschreiben folgende Leistungsmerkmale:

- Ausbau der CPU
- Kommunikation

Diagnosezustandsdaten

Diagnosezustandsdaten beschreiben den aktuellen Zustand der Komponenten, die durch die Systemdiagnose überwacht werden.

Diagnosedaten auf Baugruppen

Die einer CPU zugeordneten diagnosefähigen Baugruppen besitzen Diagnosedaten, die auf den Baugruppen selbst abgelegt sind.

Diagnosepuffer

Der Diagnosepuffer enthält Diagnoseeinträge in der Reihenfolge ihres Auftretens.

D.2 Aufbau einer SZL-Teilliste

Grundlagen

Sie können eine Teilliste oder einen Teillistenauszug mit dem Aufruf des M7-API `M7SZLRead` auslesen.

Aufbau

Eine Teilliste besteht aus

- einem Kopf und
- den Datensätzen.

Kopf

Der Kopf einer Teilliste besteht aus:

- SZL-ID
- Index
- Länge eines Datensatzes dieser Teilliste in Byte
- Anzahl der Datensätze, die diese Teilliste enthält

Index

Für bestimmte Teillisten bzw. Teillistenauszüge ist die Angabe einer Objekttypkennung bzw. einer Objektnummer notwendig. Dazu wird der Index benutzt. Wird er bei einer Auskunft nicht benötigt, so ist sein Inhalt irrelevant.

Datensätze

Ein Datensatz einer Teilliste hat eine bestimmte Länge. Sie hängt davon ab, welche Informationen in der Teilliste abgelegt sind. Wie die Datenworte in einem Datensatz belegt sind, hängt ebenfalls von der Teilliste ab.

D.3 SZL-ID

SZL-ID

Jede Teilliste innerhalb der SZL besitzt eine Nummer. Die Ausgabe einer Teilliste kann komplett oder partiell angefordert werden. Die möglichen Teillistenauszüge sind fest definiert und werden ihrerseits durch eine Nummer gekennzeichnet. Die SZL-ID setzt sich zusammen aus der Nummer der Teilliste und der Nummer des Teillistenauszugs und einer Baugruppentypklasse.

Aufbau

Die SZL-ID ist ein Wort lang. Sie ist wie folgt zusammengesetzt:

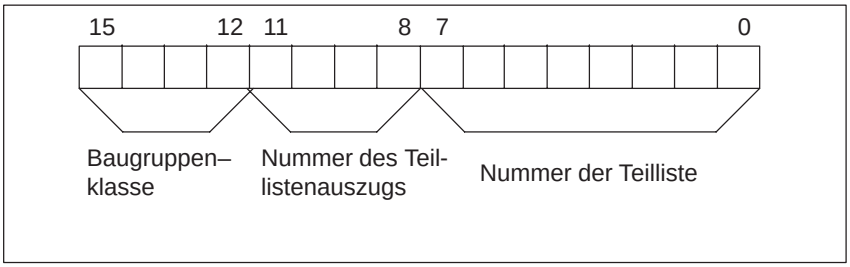

Bild D-1 Aufbau der SZL-ID

Baugruppenklasse

Beispiele für Baugruppentypen:

Baugruppentyp	Baugruppentypklasse (binär)
CPU	0000
IM	0100
FM	1000
CP	1100

Nummer des Teillistenauszugs

Die Nummern der Teillistenauszüge und ihre Bedeutung sind abhängig von der jeweiligen Teilliste. Über die Nummer des Teillistenauszugs geben Sie an, welche Untermenge einer Teilliste Sie lesen möchten.

Nummer der Teilliste

Über die Nummer der Teilliste geben Sie an, welche Teilliste der SZL Sie auslesen möchten.

D.4 Mögliche SZL-Teillisten

Subset

In einer Baugruppe ist immer nur eine Untermenge aller möglichen Teillisten verfügbar. Welche Teillisten verfügbar sind, hängt von der Baugruppe ab.

mögliche SZL- Teillisten

In der folgenden Tabelle sind alle möglichen Teillisten mit zugehöriger Nummer in der SZL-ID dargestellt.

Teilliste	SZL-ID
Baugruppen-Identifikation	W#16#xy11
CPU-Merkmale	W#16#xy12
Anwenderspeicherbereiche	W#16#xy13
Systembereiche	W#16#xy14
Bausteintypen	W#16#xy15
Alarmstatus	W#16#xy22
Betriebszustände	W#16#xy24
Kommunikation: Zustandsdaten	W#16#xy32
Baugruppenzustandsinfo	W#16#xy91
Baugruppenträger- / Stationszustandsinformation	W#16#xy92
Diagnosepuffer in der CPU	W#16#xyA0
Baugruppendiagnoseinfo (DS 0)	W#16#xyB1
Baugruppendiagnoseinfo (DS 1), geographische Adresse	W#16#xyB2
Baugruppendiagnoseinfo (DS 1), logische Adresse	W#16#xyB3
Diagnosedaten eines DP-Slaves	W#16#00B4

D.5 SZL-ID W#16#xy11 - Baugruppen-Identifikation

Zweck

Über die Teilliste mit der SZL-ID W#16#xy11 erhalten Sie die Baugruppen-Identifikation der Baugruppe.

Kopf

Der Kopf der SZL mit der SZL-ID W#16#xy11 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0011: alle Identifikationsdatensätze einer Baugruppe W#16#0111: ein einzelner Identifikationsdatensatz W#16#0F11: nur SZL-Teillistenkopfinfo
INDEX	Nur bei SZL-ID W#16#0111: Nummer eines bestimmten Datensatzes W#16#0001: Identifikation der Baugruppe W#16#0002: Identifikation der Firmware W#16#0003: Identifikation der Hardware
LENGTHDR	W#16#001C: 1 Datensatz ist 14 Worte lang (28 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der SZL mit der SZL-ID W#16#xy11 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	Nummer eines Identifikations-Datensatzes
MlfB	10	MLFB der Baugruppe; String aus 19 Zeichen und einem Blank (20H); z.B. für CPU 314: ”6ES7 314-0AE01-0AB0 ”
BGTyp	1	Baugruppentypkennung (Die Baugruppentypkennung wird nur für interne Zwecke verwendet.)
Ausbg	1	Ausgabestand der Baugruppe bzw. des Betriebssystems
Ausbe	1	Ausgabestand des PG-Beschreibungsfiles

Erweiterungen für M7–300/400

Für die Baugruppen des M7–300/400 sind die Indices > 1 reserviert. Die Indices >1 sind optional, d.h. Nichtvorhandensein bestimmter Indices (bei M7–SYS V 1.x) ist kein Fehler. Die Erweiterungen haben folgende Bedeutung:

	Index = 1 Identifikation der Baugruppe	Index = 2 Identifikation der Firmware	Index = 3 Identifikation der Hardware
MlfB	MLFB der Baugruppe	MLFB der Systemsoftware M7–SYS	MLFB der Baugruppe
BGTyp	Baugruppentypkennung	Baugruppentypkennung	Baugruppentypkennung
Ausbg	Version der Systemsoftware M7–SYS (nur die ersten zwei Stellen, dezimal) z.B. 20 für 2.0.	Vollständige Version der Systemsoftware M7–SYS (BCD codiert, vier Stellen)	Die ersten zwei Stellen beinhalten die Firmware-Version, die letzten zwei Stellen den Hardware–Erzeugnisstand, beides BCD codiert.
Ausbe	–	–	BIOS–Version in ASCII.

D.6 SZL-ID W#16#xy12 - CPU-Merkmale

Zweck

Baugruppen vom Baugruppentyp CPU besitzen eine Reihe von Merkmalen mit starkem Hardwarebezug. Jedem dieser Merkmale ist eine Kennung zugeordnet. Über die Teilliste mit der SZL-ID W#16#xy12 erhalten Sie die Merkmale der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy12 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs: W#16#0012: alle Merkmale W#16#0112: Merkmale einer Gruppe Die Gruppe geben Sie im Parameter INDEX an. W#16#0F12: nur SZL-Teillistenkopfinfo
INDEX	Gruppe W#16#0100: Zeitsystem W#16#0200: Systemverhalten

Inhalt	Bedeutung
LENGTHDR	W#16#0002: 1 Datensatz ist 1 Wort lang (2 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy12 ist ein Wort lang. Für jedes vorhandene Merkmal ist eine Merkmalkennung abgelegt. Eine Merkmalkennung ist 1 Wort lang.

Merkmalkennung

In der folgenden Tabelle sind alle Merkmalkennungen aufgeführt.

Kennung	Bedeutung
W#16#0100 - 01FF Zeitsystem (Gruppe mit Index 0100)	
W#16#0101	1-ms-Zeitauflösung
W#16#0102	10ms-Zeitauflösung
W#16#0103	keine Echtzeituhr
W#16#0104	BCD-Uhrzeitformat
W#16#0200 - 02FF Systemverhalten (Gruppe mit Index 0200)	
W#16#0201	Multicomputingfähig

D.7 SZL-ID W#16#xy13 - Anwenderspeicherbereiche

Zweck

Über die Teilliste mit der SZL-ID W#16#xy13 erhalten Sie Informationen über die Speicherbereiche der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy13 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0013: Datensätze aller Speicherbereiche W#16#0113: Datensatz für einen Speicherbereich Den Speicherbereich geben Sie über den Parameter INDEX an. W#16#0F13: nur SZL-Tellistenkopfinfo
INDEX	Angabe eines Speicherbereiches (nur bei SZL-ID W#16#0113) W#16#0001: Arbeitsspeicher
LENGTHDR	W#16#0024: 1 Datensatz ist 18 Worte lang (36 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy13 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	Index eines Speicherbereichs W#16#0001: Arbeitsspeicher
Code	1	Speichertyp: W#16#0001: flüchtiger Speicher (RAM) W#16#0002: Nicht flüchtiger Speicher (FEPROM) W#16#0003: gemischter Speicher (RAM und FEPROM)
Größe	2	Gesamtgröße des selektierten Speichers (Summe von Ber 1 und Ber 2)
Modus	1	logischer Modus des Speichers Bit 0: flüchtiger Speicherbereich Bit 1: nichtflüchtiger Speicherbereich Bit 2: gemischter Speicherbereich Für Arbeitsspeicher: Bit 3: Code und Daten getrennt Bit 4: Code und Daten gemeinsam

Name	Länge in Worten	Bedeutung
Granu	1	stets mit 0 belegt
Ber1	2	Größe des flüchtigen Speicherbereichs in Byte
Belegt1	2	Größe des belegten flüchtigen Speicherbereichs
Block1	2	Größter freier Block im flüchtigen Speicherbereich Falls 0: keine Information vorhanden oder Information nicht ermittelbar.
Ber2	2	Größe des nichtflüchtigen Speicherbereichs in Byte
Belegt2	2	Größe des belegten nichtflüchtigen Speicherbereichs
Block2	2	Größter freier Block im nichtflüchtigen Speicherbereich Falls 0: keine Information vorhanden oder Information nicht ermittelbar.

D.8 SZL-ID W#16#xy14 - Systembereiche

Zweck

Über die Teilliste mit der SZL-ID W#16#xy14 erhalten Sie Informationen über die Systembereiche der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy14 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0014: alle Systembereiche einer Baugruppe W#16#0114: ein Systembereich, Angabe über Parameter INDEX W#16#0F14: nur SZL-Teillistenkopfinfo
INDEX	Nur für SZL-ID W#16#0114: W#16#0001: PAE (Anzahl in Byte) W#16#0002: PAA (Anzahl in Byte) W#16#0003: Merker (Anzahl) W#16#0004: Zeiten (Anzahl) W#16#0005: Zähler (Anzahl) W#16#0006: Anzahl der Byte im logischen Adreßraum (Anzahl) W#16#0007: Größe des gesamten Lokaldatenbereichs der CPU in Byte
LENGTHDR	W#16#0008: 1 Datensatz ist 4 Worte lang (8 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy14 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	Index des Systembereichs W#16#0001: PAE (Anzahl in Byte) W#16#0002: PAA (Anzahl in Byte) W#16#0003: Merker (Anzahl in Bit) W#16#0004: Zeiten (Anzahl) W#16#0005: Zähler (Anzahl) W#16#0006: Anzahl der Byte im logischen Adreßraum W#16#0007: Lokaldaten (gesamter Lokaldatenbereich der Baugruppe in Byte)
code	1	Speichertyp W#16#0001: flüchtiger Speicher (RAM) W#16#0002: Nicht flüchtiger Speicher (FEPROM) W#16#0003: gemischter Speicher (RAM und FEPROM)
an-zahl	1	Anzahl der Elemente des Systembereichs
reman	1	Anzahl der remanenten Elemente

D.9 SZL-ID W#16#xy15 - Bausteintypen

Zweck

Über die Teilliste mit der SZL-ID W#16#xy15 erhalten Sie die Bausteintypen, die in einer Baugruppe vorhanden sind.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy15 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0015: Datensätze aller Bausteintypen einer Baugruppe W#16#0115: Datensatz eines Bausteintyps Den Bausteintyp geben Sie über den Parameter INDEX an. W#16#0F15: nur SZL-Teillistenkopfinfo
INDEX	Nur für SZL-ID W#16#0115: W#16#0800: OB W#16#0A00: DB W#16#0B00: SDB W#16#0C00: FC W#16#0E00: FB
LENGTHDR	W#16#000A: 1 Datensatz ist 5 Worte lang (10 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy15 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	Bausteintyp-Nummer W#16#0800: OB W#16#0A00: DB W#16#0B00: SDB W#16#0C00: FC W#16#0E00: FB
MaxAnz	1	maximale Anzahl der Bausteine des Typs bei OBs: max. mögliche Anzahl OBs einer CPU bei DBs: max. mögliche Anzahl DBs einschließlich DB0 bei SDBs: max. mögliche Anzahl SDBs einschließlich SDB2 bei FCs und FBs: max. mögliche Anzahl ladbarer Bausteine
MaxLng	1	maximale Gesamtgröße des Ladeobjekts in KByte
Maxabl	2	maximale Länge des Arbeitsspeicheranteils eines Bausteins in Byte

D.10 SZL-ID W#16#xy22 - Alarmstatus

Zweck

Über die Teilliste mit der SZL-ID W#16#xy22 erhalten Sie Informationen über den aktuellen Zustand der Alarmbearbeitung und der Alarmgenerierung der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy22 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0222: Datensatz zum angegebenen Alarm. Den Alarm geben Sie im Parameter INDEX an. W#16#0F22: nur SZL-Teillistenkopfinfo
INDEX	Alarmklasse oder OB-Nr. (bei SZL-ID W#16#0222) W#16#0001: freier Zyklus (OB Nr. 1) W#16#5050: Asynchroner Fehleralarm (OB Nr. 80)
LENGTHDR	W#16#001C: 1 Datensatz ist 14 Worte lang (28 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy22 hat folgenden Aufbau:

Name	Länge	Bedeutung
info	10 Worte	Startinfo des jeweiligen OBs, mit folgenden Ausnahmen: • Beim OB 1 sind die aktuelle minimale und die maximale Zykluszeit zu entnehmen. • Beim OB 80 sind die projektierte minimale und die projektierte maximale Zykluszeit lesbar.
al1 bis 3	4 Worte	reserviert

D.11 SZL-ID W#16#xy24 - Betriebszustand und Betriebszustandsübergang

Zweck

Über die Teilliste mit der SZL-ID W#16#xy24 erhalten Sie Informationen über die Betriebszustände der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy24 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0024: alle Betriebszustände, die in der Baugruppe gespeichert sind W#16#0124: Informationen über den letzten durchgeführten Betriebszustandsübergang W#16#0424: aktueller Betriebszustand W#16#0524: angegebener Betriebszustand Den Betriebszustand geben Sie über den Parameter INDEX an W#16#0F24: nur SZL-Teillistenkopfinfo
INDEX	nur für SZL-ID W#16#0524: Betriebszustand W#16#5000: Betriebszustand STOP W#16#5010: Betriebszustand ANLAUF W#16#5020: Betriebszustand RUN W#16#5030: Betriebszustand HALT W#16#4520: Betriebszustand DEFEKT
LENGTHDR	W#16#0014: 1 Datensatz ist 10 Worte lang (20 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xy24 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Info	10	BZ-Info oder BZÜ-Info

BZÜ-Info

Die Information über einen Betriebszustandsübergang (BZÜ-Info) ist 20 Byte lang und wie folgt aufgebaut:

Name	Länge	Bedeutung
ereig	1 Wort	Ereignis-ID möglich W#16#4xy?
ae	1 Byte	B#16#FF
bzü-id	1 Byte	ID des Betriebzustandübergangs (BZÜ-ID) aufgeteilt in je 4 Bit Bit 0 bis 3: angeforderter Betriebszustand (BZ) Bit 4 bis 7: bisheriger Betriebszustand (BZ) BZ-IDs der angeforderten bzw. bisherigen Betriebszustände: 1H: STOP (update) 2H: STOP (Urlöschen) 3H: STOP (Eigeninitialisierung) 4H: STOP (intern) 6H. Anlauf (Neustart) 7H: Wiederanlauf 8H: RUN AH: HALT DH: DEFEKT
res	2 Worte	reserviert
anlinfo 1	1 Byte	Bit 0: Anlaufzusatzkennung = 0: Anlauf ohne geänderten Systemausbau = 1: Anlauf mit geändertem Systemausbau Bits 5 und 4: Multicomputing–Verhalten = 00B: Einzelprozessorbetrieb = 01B: Multicomputingbetrieb
anlinfo 2	1 Byte	B#16#01: automatischer Neustart bei Multicomputingbetrieb B#16#03: manueller Neustart über Betriebsartenschalter B#16#04: Neustart über Kommunikationsfunktion B#16#0A: automatischer Wiederanlauf bei Multicomputingbetrieb B#16#0B: manueller Wiederanlauf über Betriebsartenschalter B#16#0C: Wiederanlauf über Kommunikationsfunktion B#16#10: gepufferter automatischer Neustart B#16#20: ungepufferter automatischer Neustart B#16#A0: gepufferter automatischer Wiederanlauf

Name	Länge	Bedeutung
anlinfo 3	1 Byte	Unzulässigkeit bestimmter Anlaufarten B#16#?0: manueller Anlauf unzulässig (Urlöschen angefordert) B#16#?1: manueller Anlauf unzulässig, Parameteränderung etc. ist erforderlich B#16#?7: manueller Neustart zulässig B#16#?F: manueller Neustart und manueller Wiederanlauf zulässig B#16#0?: automatischer Anlauf unzulässig B#16#1?: automatischer Anlauf unzulässig, Parameteränderung etc. ist erforderlich B#16#3?: ungepufferter, automatischer Neustart unzulässig B#16#7?: automatischer Neustart zulässig B#16#F?: automatischer Neustart und automatischer Wiederanlauf zulässig
anlinfo 4	1 Byte	letzte gültige Bedienung oder Einstellung B#16#?0 keine Bedienung B#16#?1: Neustart bei Multicomputingbetrieb B#16#?3: manueller Neustart über Betriebsartenschalter B#16#?4: Neustart über Kommunikationsfunktion B#16#?A: Wiederanlauf bei Multicomputingbetrieb B#16#?B: manueller Wiederanlauf über Betriebsartenschalter B#16#?C: Wiederanlauf über Kommunikationsfunktion B#16#0? keine Bedienung B#16#1?: gepufferter automatischer Neustart B#16#2?: ungepufferter automatischer Neustart B#16#A?: automatischer Wiederanlauf, parametriert
time	4 Worte	Zeitstempel

BZ-Info

Die Information über einen Betriebszustand (BZ-Info) ist 20 Byte lang und wie folgt aufgebaut:

Name	Länge	Bedeutung
ereig	1 Wort	Ereignis-ID möglich W#16#5xy?
ae	1 Byte	B#16#FF
bz-id	1 Byte	BZ-ID: B#16#01: STOP (update) B#16#02: STOP (Urlöschen) B#16#03: STOP (Eigeninitialisierung) B#16#04: STOP (intern) B#16#06: Anlauf (Neustart) B#16#07: Anlauf (Wiederanlauf) B#16#08: RUN B#16#0A: HALT B#16#0D: DEFEKT

Name	Länge	Bedeutung
res	4 Worte	reserviert
time	4 Worte	Zeitstempel

D.12 SZL-ID W#16#xy32 - Kommunikationszustandsdaten

Zweck

Über die Teilliste mit der SZL-ID W#16#xy32 erhalten Sie die Kommunikationszustandsdaten der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy32 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0132: Zustandsdaten zu einem Kommunikationsteil Den Kommunikationsteil geben Sie im Parameter INDEX an. W#16#0F32: nur SZL-Teillistenkopfinfo
INDEX	Nur für SZL-ID W#16#0132: Kommunikationsteil W#16#0001: Allgemeine Kommunikationsdaten W#16#0005: Diagnose W#16#0008: Zeitsystem
LENGTHDR	W#16#00028: 1 Datensatz ist 20 Worte lang (40 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#0132 ist immer 20 Worte lang. Die Datensätze sind unterschiedlich belegt. Die Belegung hängt vom Parameter INDEX ab, d. h. davon, zu welchem Kommunikationsteil der Datensatz gehört.

D.13 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0001

Inhalt

Der Teillistenauszug mit der SZL-ID W#16#0132 und dem Index W#16#0001 enthält allgemeine Kommunikationszustandsdaten.

Datensatz

Ein Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 und dem Index W#16#0001 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	W#16#0001: allgemeine Zustandsdaten zur Kommunikation
res pg	1	garantierte Anzahl von PG-Verbindungen
res os	1	garantierte Anzahl von OS-Verbindungen
u pg	1	aktuelle Anzahl der PG-Verbindungen
u os	1	aktuelle Anzahl der OS-Verbindungen
proj	1	aktuelle Anzahl der festprojektierten Verbindungen
auf	1	aktuelle Anzahl der von proj aufgebauten Verbindungen
free	1	Anzahl der freien Verbindungen
used	1	Anzahl der benutzten freien Verbindungen
last	1	maximal eingestellte Kommunikationlast der CPU in %
res	10	reserviert

D.14 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0005

Inhalt

Der Teillistenauszug mit der SZL-ID W#16#0132 und dem Index W#16#0005 enthält Informationen über den Diagnosezustand der Baugruppe.

Datensatz

Ein Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 und dem Index W#16#0005 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	W#16#0005: Diagnose
erw	1	erweiterter Funktionsumfang 0: nein 1: ja
send	1	Automatisches Senden 0: nein 1: ja
moeg	1	Senden von Anwenderdiagnoseeinträgen derzeit möglich 0: nein 1: ja
res	15	reserviert

D.15 Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 Index W#16#0008

Inhalt

Der Teillistenauszug mit der SZL-ID W#16#0132 und dem Index W#16#0008 enthält Informationen über den Zustand des Zeitsystems der Baugruppe.

Datensatz

Ein Datensatz des Teillistenauszugs mit der SZL-ID W#16#0132 und dem Index W#16#0008 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
Index	1	W#16#0008: Zeitsystem-Zustand
zykl	1	Zykluszeit der Synchronisationstelegramme
korr	1	Korrekturfaktor für die Uhrzeit
clock 0	1	Betriebsstundenzähler 0: Zeit in Stunden
clock 1	1	Betriebsstundenzähler 1: Zeit in Stunden
clock 2	1	Betriebsstundenzähler 2: Zeit in Stunden
clock 3	1	Betriebsstundenzähler 3: Zeit in Stunden
clock 4	1	Betriebsstundenzähler 4: Zeit in Stunden
clock 5	1	Betriebsstundenzähler 5: Zeit in Stunden
clock 6	1	Betriebsstundenzähler 6: Zeit in Stunden
clock 7	1	Betriebsstundenzähler 7: Zeit in Stunden
time	4	aktuelles Datum und Uhrzeit (Format: date_and_time)
res	5	reserviert

D.16 SZL-ID W#16#xy91 - Baugruppenzustandsinformation

Zweck

Über die Teilliste mit der SZL-ID W#16#xy91 erhalten Sie die Zustandsinformation über Baugruppen.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy91 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs W#16#0991: Zustandsinformation aller Stationen eines DP–Mastersystems W#16#0A91: Zustandsinformation aller PROFIBUS–Subnetze und DP–Mastersysteme W#16#0C91: Baugruppenzustandsinformation einer Baugruppe über die logische Basisadresse W#16#0D91: Baugruppenzustandsinformation aller Baugruppen im angegebenen Baugruppenträger W#16#0F91: nur SZL-Teillistenkopfinfo
INDEX	• Für den Teillistenauszug mit der SZL-ID W#16#0C91: Bit 0 bis 14: logische Basisadresse der Baugruppe Bit 15: 0 = Eingang, 1 = Ausgang • Für den Teillistenauszug mit den SZL-IDs W#16#0991: HighByte: Subsystem-ID, LowByte: 0 • Für den Teillistenauszug mit den SZL-IDs W#16#0D91: W#16#00xx alle Baugruppen und Submodule eines Baugruppenträgers (xx enthält die Nummer des Baugruppenträgers)
LENGTHDR	W#16#0010: 1 Datensatz ist 8 Worte lang (16 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der ID W#16#xy91 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
geoadr1	1	Nummer des Baugruppenträgers (Subnet-ID und Stationsnummer bei DP) der Geoadresse Nr. des Baugruppenträgers bei W#16#0A91
geoadr2	1	Steckplatz und Submodulsteckplatz
logadr	1	erste zugeordnete logische E/A-Adresse (Basisadresse)
solltyp	1	Soll-Baugruppentyp
isttyp	1	Ist-Baugruppentyp

Name	Länge in Worten	Bedeutung
alarm	1	reserviert
eastat	1	E/A-Status Bit 0 = 1: Baugruppe gestört (über Diagnosealarm erkannt) Bit 1 = 1: Baugruppe vorhanden Bit 2 = 1: Baugruppe nicht verfügbar (durch Zugriffsfehler erkannt) Bit 5 = 1: Baugruppe kann Hostbaugruppe für Submodule sein Bit 6 = 1: reserviert für S7-400 Bit 7 = 1: Baugruppe im Lokalbussegment Bit 8 bis 15: Datenkennung für logische Adresse (Eingang: B#16#B4, Ausgang: B#16#B5, externe DP-Anschaltung: B#16#FF)
ber_bgbr	1	Bereichskennung/Baugruppenbreite Bit 0 bis 2 : Baugruppenbreite Bit 4 bis 6 : Bereichskennung 0 = M7-400 1 = M7-300 2 = ET-Bereich 3 = P-Bereich 4 = Q-Bereich 5 = IM3-Bereich 6 = IM4-Bereich

geoadr1

Der Parameter geoadr1 enthält

- bei zentralem Aufbau die Baugruppenträgernummer

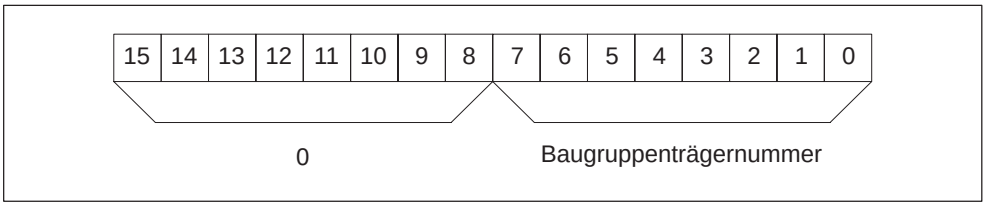

Bild D-2 Belegung des Parameters geoadr1 bei zentralem Aufbau

- bei dezentralem Aufbau die DP-Mastersystem-ID und die Stationsnummer.

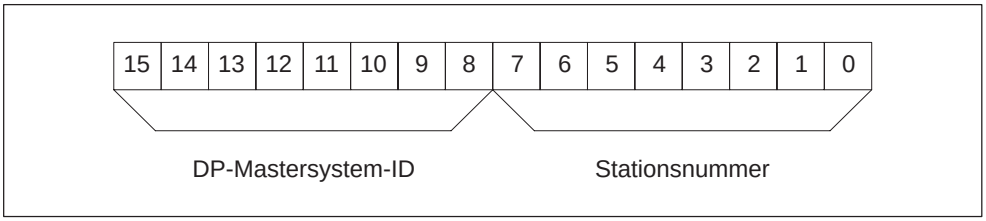

Bild D-3 Belegung des Parameters geoadr1 bei dezentralem Aufbau

geoadr2

Der Parameter geoadr2 enthält Steckplatz und Submodulsteckplatz.

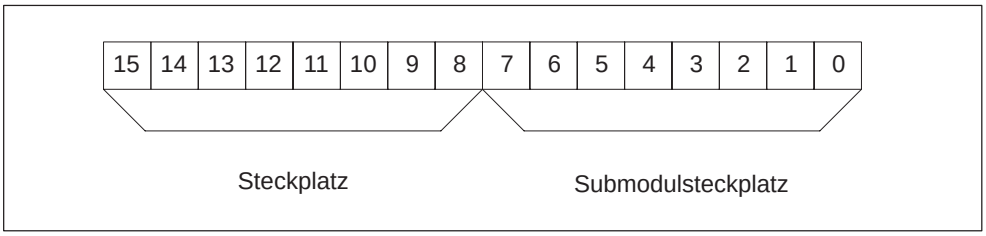

Bild D-4 Belegung des Parameters geoadr2

D.17 SZL-ID W#16#xy92 - Baugruppenträger-/Stationszustandsinformation

Zweck

Über die Teilliste mit der SZL-ID W#16#xy92 erhalten Sie Auskünfte über den Soll- und den Ist-Ausbau von Baugruppenträgern bei zentralem Aufbau und Stationen eines DP-Mastersystems.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xy92 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs: W#16#0092: Sollzustand der Baugruppenträger im zentralen Aufbau/ der Stationen eines DP-Mastersystems W#16#0292: Istzustand der Baugruppenträger im zentralen Aufbau/ der Stationen eines DP-Mastersystems W#16#0692: OK–Zustand der Erweiterungsgeräte im zentralen Aufbau/ der Stationen eines DP–Mastersystems, das über eine integrierte DP–Anschaltung angeschlossen ist W#16#0F92: Nur SZL-Teillistenkopfinfo
INDEX	=0: Informationen über Baugruppenträger im zentralen Aufbau ≠0 (=DP-Mastersystem-ID): Informationen über Stationen im DP-Bereich
LENGTHDR	W#16#0010: Datensatz ist acht Worte lang (16 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der ID W#16#xy92 hat folgenden Aufbau:

Inhalt	Länge	Bedeutung
status	8 Worte	Rackstatus/ Stationsstatus oder Pufferungsstatus. Es gibt folgende Zuordnung der Bits zu den Baugruppenträgern: Bit 0: Zentralgerät Bit 1-21: 1. bis 21. Erweiterungsgerät Bit 22-29: stets 0 Bit 30: Erweiterungsgerät-Ausfall im SIMATIC-S5-Bereich Bit 31 bis 127: stets 0 Falls Sie INDEX ungleich 0 gewählt haben, gilt folgende Zuordnung zwischen Bits und den Stationen (DP–Mastersystem): Bit 0: Station 1 Bit 1: Station 2 : : Bit 127: Station 128 W#16#0092: Bit=0: Baugruppenträger/-Station nicht projektiert Bit=1: Baugruppenträger/-Station projektiert W#16#0292: Bit=0: Baugruppenträger/-Station ausgefallen oder nicht projektiert Bit=1: Baugruppenträger/-Station vorhanden und nicht ausgefallen W#16#0692: Bit=0: alle Baugruppen des Erweiterungsgeräts/ einer Station sind vorhanden, verfügbar und nicht gestört Bit=1: mindestens eine Baugruppe des Erweiterungsgeräts/einer Station ist nicht OK

D.18 SZL-ID W#16#xyA0 - Diagnosepuffer

Zweck

Über die Teilliste mit der SZL-ID W#16#xyA0 erhalten Sie die Einträge im Diagnosepuffer der Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#xyA0 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	Die SZL-ID des Teillistenauszugs: W#16#00A0: alle im aktuellen Betriebszustand lieferbaren Einträge W#16#01A0: die neuesten Eintragungen; die Anzahl der neuesten Eintragungen geben Sie über den Parameter INDEX an W#16#0DA0: alle Diagnoseeinträge (nur im STOP) W#16#0EA0: alle Anwendereinträge(nur im STOP). Mit dem Aufruf *M7WriteDiagnose* können Sie eigene Einträge im Bereich A000H bis BFFFH definieren. W#16#0FA0: nur SZL-Teillistenkopfinfo
INDEX	Nur für SZL-ID W#16#01A0: Anzahl der neuesten Eintragungen
LENGTHDR	W#16#0014: 1 Datensatz ist 10 Worte lang (20 Byte)
N_DR	Anzahl der Datensätze

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#xyA0 hat folgenden Aufbau:

Name	Länge in Worten	Bedeutung
ID	1	Ereignis-ID
info	5	Informationen zum Ereignis bzw. zu dessen Wirkung
time	4	Zeitstempel des Ereignisses

Diagnosepuffer

Weitere Informationen zu Ereignissen im Diagnosepuffer erhalten Sie über STEP 7.

D.19 SZL-ID W#16#00B1 - Baugruppendiagnoseinfo

Zweck

Über die Teilliste mit der SZL-ID W#16#00B1 erhalten Sie die ersten 4 Diagnosebytes (Diagnosedatensatz DS0) einer diagnosefähigen Baugruppe.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#00B1 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	W#16#00B1
INDEX	Bit 0 bis 14: logische Basisadresse Bit 15: 0 für Eingang, 1 für Ausgang
LENGTHDR	W#16#0004: 1 Datensatz ist 2 Worte lang (4 Byte)
N_DR	1

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#00B1 hat folgenden Aufbau:

Name	Länge	Bedeutung	
byte1	1 Byte	Bit 0:	Baugruppenstörung/OK (Sammelstörkennung)
		Bit 1:	Fehler intern
		Bit 2:	Fehler extern
		Bit 3:	Kanalfehler vorhanden
		Bit 4:	externe Hilfsspannung fehlt
		Bit 5:	Frontstecker fehlt
		Bit 6:	Baugruppe nicht parametriert
		Bit 7:	Falsche Parameter in Baugruppe
byte2	1 Byte	Bit 0 bis Bit 3:	Baugruppenklasse (CPU, FM, CP, IM, SM, ...)
		Bit 4:	Kanalinformation vorhanden
		Bit 5:	Anwenderinformation vorhanden
		Bit 6:	Diagnosealarm von Stellvertreter
		Bit 7:	Reserve (mit 0 initialisiert)

Name	Länge	Bedeutung	
byte3	1 Byte	Bit 0:	Anwendermodul falsch/fehlt
		Bit 1:	Kommunikationsstörung
		Bit 2:	Betriebszustand RUN/STOP (0 = RUN, 1 = STOP)
		Bit 3:	Zeitüberwachung angesprochen (watch dog)
		Bit 4:	BG-interne Versorgungsspannung ausgefallen
		Bit 5:	Batterie leer (BFS)
		Bit 6:	Gesamte Pufferung ausgefallen
		Bit 7:	Reserve (mit 0 initialisiert)
byte4	1 Byte	Bit 0:	Rackausfall (von IM erkannt)
		Bit 1:	Prozessorausfall
		Bit 2:	EPROM-Fehler
		Bit 3:	RAM-Fehler
		Bit 4:	ADU/DAU-Fehler
		Bit 5:	Sicherungsfall
		Bit 6:	Prozeßalarm verloren
		Bit 7:	Reserve (mit 0 initialisiert)

D.20 SZL-ID W#16#00B2 - Baugruppendiagnosedaten über geographische Adresse

Zweck

Über die Teilliste mit der SZL-ID W#16#00B2 erhalten Sie alle Diagnosedaten einer zentral gesteckten Baugruppe (also nicht für DP und Submodule). Die Baugruppe geben Sie über Baugruppenträger und Steckplatznummer an.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#00B2 ist wie folgt aufgebaut:

Inhalt	Bedeutung	
SZL-ID	W#16#00B2	
INDEX	W#16#xxyy:	xx enthält die Nummer des Baugruppenträgers yy enthält die Steckplatznummer
LENGTHDR	Länge des Datensatzes hängt von der Baugruppe ab	
N_DR	1	

Datensatz

Wie groß ein Datensatz der Teilliste mit der SZL-ID W#16#00B2 ist und wie der Datensatz belegt ist, hängt von der jeweiligen Baugruppe ab. Nähere Informationen hierzu finden Sie in Anhang E oder im jeweiligen Handbuch zur Baugruppe.

D.21 SZL-ID W#16#00B3 - Baugruppendiagnosedaten über logische Basisadresse

Zweck

Über die Teilliste mit der SZL-ID W#16#00B3 erhalten sie alle Diagnosedaten (Diagnosedatensatz DS1) einer Baugruppe. Diese Auskunft ist auch für DP und Submodule möglich. Sie wählen die Baugruppe über ihre logische Basisadresse aus.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#00B3 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	W#16#00B3
INDEX	Bit 0 bis 14: logische Basisadresse Bit 15: 0 für Eingang, 1 für Ausgang
LENGTHDR	Länge eines Datensatzes (baugruppenabhängig)
N_DR	1

Datensatz

Wie groß ein Datensatz der Teilliste mit der SZL-ID W#16#00B3 ist und wie er belegt ist, hängt von der jeweiligen Baugruppe ab. Nähere Informationen hierzu finden Sie in Anhang E oder im jeweiligen Handbuch zur Baugruppe.

D.22 SZL-ID W#16#00B4 – Diagnosedaten eines DP-Slaves

Zweck

Über die Teilliste mit der SZL-ID W#16#00B4 erhalten Sie die Diagnosedaten eines DP-Slaves. Diese Diagnosedaten sind nach EN50 170 Volume 2, PROFIBUS aufgebaut. Sie wählen die Baugruppe über ihre projektierte Diagnoseadresse aus.

Kopf

Der Kopf der Teilliste mit der SZL-ID W#16#00B4 ist wie folgt aufgebaut:

Inhalt	Bedeutung
SZL-ID	W#16#00B4
INDEX	Projektierte Diagnoseadresse des DP-Slaves
LENGTHDR	Länge eines Datensatzes. Die Maximallänge beträgt 240 Byte; bei Normslaves, bei denen die Anzahl der Normdiagnosedaten größer als 240 Byte ist und maximal 244 Byte beträgt, werden die ersten 240 Byte gelesen und das entsprechende Overflow-Bit in den Daten gesetzt.
N_DR	1

Datensatz

Ein Datensatz der Teilliste mit der SZL-ID W#16#00B4 hat folgenden Aufbau:

Name	Länge	Bedeutung
status1	1 Byte	Stationsstatus1
status2	1 Byte	Stationsstatus2
status3	1 Byte	Stationsstatus3
stat_nr	1 Byte	Master-Stationsnummer
ken_hi	1 Byte	Herstellerkennung (high byte)
ken_lo	1 Byte	Herstellerkennung (low byte)
....	....	Weitere slavespezifische Diagnose

Diagnosedaten

E

Einführung

Dieser Anhang beschreibt den Aufbau und Inhalt der Diagnosedaten für M7–Baugruppen.

Datensatz 0 und 1 der Systemdaten

Die Diagnosedaten einer Baugruppe stehen in den Datensätzen 0 und 1 des Systemdatenbereichs:

- Der Datensatz 0 enthält 4 Byte Diagnosedaten, die den aktuellen Zustand einer Signalbaugruppe oder Funktionsbaugruppe beschreiben.
- Der Datensatz 1 enthält
  - die 4 Byte Diagnosedaten, die auch im Datensatz 0 stehen, **und**
  - die baugruppenspezifischen Diagnosedaten.

Aufbau und Inhalt der Diagnosedaten

Im folgenden ist der Aufbau und der Inhalt der einzelnen Bytes der Diagnosedaten beschrieben.

Generell gilt: Wenn ein Fehler auftritt, dann wird das entsprechende Bit auf ”1” gesetzt.

Tabelle E-1 Aufbau und Inhalt der Diagnosedaten

Byte	Bit	Bedeutung	Bemerkung	
0	0	Baugruppenstörung		
	1	Fehler intern		
	2	Fehler extern		
	3	Kanalfehler vorhanden		
	4	Externe Hilfsspannung fehlt		
	5	Frontstecker fehlt		
	6	Parametrierung fehlt		
	7	Falsche Parameter in der Baugruppe		
1	0 bis 3	Baugruppenklasse	0101 0000 1000 1100 1111	Analogbaugruppe CPU Funktionsbaugruppe CP Digitalbaugruppe
	4	Kanalinformation vorhanden		
	5	Anwenderinformation vorhanden		
	6	Diagnosealarm von Stellvertreter		
	7	0		
2	0	Speichermodul falsch oder fehlt		
	1	Kommunikationsstörung		
	2	Betriebszustand	0 1	RUN STOP
	3	Zykluszeitüberwachung angesprochen		
	4	Baugruppeninterne Versorgungsspannung ausgefallen		
	5	Batterie leer		
	6	Gesamte Pufferung ausgefallen		
	7	0		
3	0	Baugruppenträgerausfall		
	1	Prozessorausfall		
	2	EPROM-Fehler		
	3	RAM-Fehler		
	4	ADU/DAU-Fehler		
	5	Sicherungsausfall		
	6	Prozeßalarm verloren		
	7	0		

Tabelle E-1 Aufbau und Inhalt der Diagnosedaten, Fortsetzung

Byte	Bit	Bedeutung	Bemerkung	
4	0 bis 6	Kanaltyp	B#16#70 B#16#71 B#16#72 B#16#73 B#16#74 B#16#75 B#16#76 B#16#77 B#16#78 B#16#79 bis B#16#7D B#16#7E B#16#7F	Digitaleingabe Analogeingabe Digitalausgabe Analogausgabe FM-POS FM-REG FM-ZAEHL FM-TECHNO FM-NCU reserviert US300 reserviert
	7	Weiterer Kanaltyp vorhanden?	0 1	nein ja
5	0 bis 7	Anzahl der Diagnosebits, die eine Baugruppe pro Kanal ausgibt.		
6	0 bis 7	Anzahl der gleichartigen Kanäle einer Baugruppe	Wenn auf einer Baugruppe unterschiedliche Kanaltypen existieren, dann wird für jeden Kanaltyp die Struktur ab Byte 4 im Datensatz 1 wiederholt.	
7	0	Kanalfehler Kanal 0/ Kanalgruppe 0	Erstes Byte des Kanalfehlervektors (Die Länge des Kanalfehlervektors richtet sich nach der Kanalzahl und ist auf Bytegrenzen aufgerundet.)	
	1	Kanalfehler Kanal 1/ Kanalgruppe 1		
	2	Kanalfehler Kanal 2/ Kanalgruppe 2		
	3	Kanalfehler Kanal 3/ Kanalgruppe 3		
	4	Kanalfehler Kanal 4/ Kanalgruppe 4		
	5	Kanalfehler Kanal 5/ Kanalgruppe 5		
	6	Kanalfehler Kanal 6/ Kanalgruppe 6		
	7	Kanalfehler Kanal 7/ Kanalgruppe 7		
...	–	Kanalspezifische Fehler		

Kanalspezifische Fehler

Ab dem Byte, das unmittelbar hinter dem Kanalfehlervektor liegt, werden für jeden Kanal der Baugruppe die kanalspezifischen Fehler angezeigt. Für die Bitbelegung gilt:

- 1 = Fehler
- 0 = kein Fehler

Den Aufbau der kanalspezifischen Diagnose für die unterschiedlichen Kanaltypen finden Sie im Anhang A, “Diagnosedaten” des Handbuches “Systemsoftware für S7-300/400 – System- und Standardfunktionen”.

Kanaltyp für M7–FMs

Für Applikationsbaugruppen FM 356–4 und FM 456–4 muß der Kanaltyp (Bit 0 bis 6 von Byte 4) gleich 0x77 (FM–TECHNO) gesetzt werden.

Literaturverzeichnis

F

Fibeln und Broschüren

/09/ Fibel: *Automatisierungssystem S7-200*,
S7–200 einfach aufbauen und programmieren

/12/ Broschüre: *Automatisierungssystem S7-300*,
Aufbau und Anwendung

/13/ Broschüre: *Automatisierungssystem S7-400*,
Aufbau und Anwendung

/14/ Broschüre: *Der Automatisierungsrechner SIMATIC M7*
Aufbau und Anwendung

/20/ Broschüre: *Automatisierungssystem S7-300/400*,
Programmierung

/21/ Broschüre: *Automatisierungssystem S7/M7*,
Dezentralisieren mit PROFIBUS-DP und AS-I

/30/ Fibel: *Automatisierungssystem S7-300,*
Einfach aufbauen und programmieren

System 200

/49/ Kurzinformation: *Automatisierungssystem S7-200,*
CPU 212/214/215/216

/50/ Systemhandbuch: *Automatisierungssystem S7-200,*

S7-300

/69/ Handbuch: *Reglerbaugruppe FM 355,*
Aufbauen und Parametrieren

/70/ Handbuch: *Automatisierungssystem S7-300,*
Aufbauen, CPU-Daten

/71/ Referenzhandbuch: *Automatisierungssysteme S7-300, M7-300*
Baugruppendaten

/72/ Operationsliste: *Automatisierungssystem S7-300*
CPU 312 IFM, 314 IFM, 313,314,315-2DP

/73/ Handbuch: *Automatisierungssystem S7-300,*
Intergrierte Funktionen CPU 312 IFM/314 IFM

/74/ Handbuch: *Zählerbaugruppe FM 350,*
Aufbauen und Parametrieren

/75/ Handbuch: *Positionierbaugruppe FM 351,*
Aufbauen und Parametrieren

/76/ Handbuch: *Elektronisches Nockensteuerwerk FM 352,*
Aufbauen und Parametrieren

/77/ Handbuch: *Applikationsbaugruppe FM 356,*
Aufbauen und Inbetriebnehmen

/78/ Handbuch: *Punkt-zu-Punkt-Kopplung CP340,*
Aufbauen und Parametrieren

/80/ Handbuch: *Automatisierungssystem M7-300,*
Aufbauen, CPU-Daten

S7-400

/100/ Installationshandbuch: *Automatisierungssysteme S7-400, M7-400,* Aufbauen

/101/ Referenzhandbuch: *Automatisierungssysteme S7-400, M7-400* Baugruppendaten

/102/ Operationsliste: *Automatisierungssystem S7-400*

/103/ Handbuch: *Zählerbaugruppe FM 450,* Aufbauen und Parametrieren

/104/ Handbuch: *Positionierbaugruppe FM 451,* Aufbauen und Parametrieren

/105/ Handbuch: *Elektronisches Nockensteuerwerk FM 452,* Aufbauen und Parametrieren

/106/ Handbuch: *Applikationsbaugruppe FM 456,* Aufbauen und Inbetriebnehmen

/107/ Handbuch: *Punkt-zu-Punkt-Kopplung CP441,* Aufbauen und Parametrieren

/108/ Handbuch: *Zählermodul IF 961 CT1,* Programmieren und Parametrieren

/109/ Handbuch: *Smart Connect SC,* Aufbauen

C7 Komplettgeäte

/117/ Handbuch: *Komplettgeräte C7-626, C7-626 DP* Einbauen, Montieren, Verdrahten

/118/ Handbuch: *Komplettgeräte C7-621, C7-621 AS-i* Einbauen, Montieren, Verdrahten

/119/ Handbuch: *Komplettgeräte C7-621, C7-621 AS-i* Arbeiten mit C7

/120/ Handbuch: *Komplettgeräte C7-623, C7-624* Einbauen, Montieren, Verdrahten

/121/ Handbuch: *Komplettgeräte C7-623, C7-624* Arbeiten mit C7

/122/ Handbuch: *Komplettgeräte C7-626, C7-627* Arbeiten mit C7

ET-200

/140/ Handbuch: *Dezentrales Peripheriegerät ET 200M*

/141/ Handbuch: *Dezentrales Peripheriegerät ET 200L*

/142/ Handbuch: *Dezentrales Peripheriegerät ET 200X*

/143/ Handbuch: *Dezentrales Peripheriegerät ET 200B*

/150/ Referenz-Handbuch: Automatisierungssysteme
S7-300, M7-300, ET 200M EX-Peripheriebaugruppen

STEP 7-Basis

/230/ Benutzerhandbuch: *Basissoftware für S7,*
Konvertieren von S 5-Programmen

/231/ Benutzerhandbuch: *Basissoftware für S7und M7,*
STEP 7

/232/ Handbuch: *AWL für S7-300/400,*
Bausteine programmieren

/233/ Handbuch: *KOP für S7-300/400,*
Bausteine programmieren

/234/ Programmierhandbuch: *Systemsoftware für S7-300/400*
Programmentwurf

/235/ Referenzhandbuch: *Systemsoftware für S7-300/400*
System-und Standardfunktionen

/236/ Handbuch: *FUP für S7-300/400,*
Bausteine programmieren

/237/ *Gesamtindex,* STEP 7

STEP 7-Option

/249/ Handbuch: *CFC* Technologische Funktionen grafisch verschalten, Systemspezifischer Teil: S7/M7

/250/ Handbuch: *SCL für S7-300/400,* Bausteine programmieren

/251/ Handbuch: *GRAPH für S7-300/400,* Ablaufsteuerungen programmieren

/252/ Handbuch: *HiGraph für S7-300/400,* Zustandsgraphen programmieren

/253/ Handbuch: *C für S7-300/400,* C-Programme erstellen

/254/ Handbuch: *CFC für S7 und M7,* Technologische Funktionen graphisch verschalten, Basis-Teil

/255/ Handbuch: *SFC Process Control System PCS 7* Technologische Ablaufsteuerungen projektieren

/256/ Handbuch: *Engineering System (ES)* Anlagen technologisch und phasenübergreifend strukturieren

/257/ Handbuch: *BATCH flexible–System* Chargenprozesse automatisieren

/258/ Referenzhandbuch: *Bibliothek Basisbausteine*

/259/ Referenzhandbuch: *Bibliothek Technologische Bausteine*

/260/ Referenzhandbuch: *Bibliothek Feldgerätebausteine*

/261/ Handbuch: *Process Control System PCS 7* Systembeschreibung

/270/ Handbuch: *S7-PDIAG für S7-300/400* Prozeßdiagnose für KOP, FUP und AWL projektieren

/271/ Handbuch: *NETPRO* Netze grafisch projektieren

M7-300/400

/280/ Programmierhandbuch: *Systemsoftware für M7-300/400,*
Programmentwurf

/281/ Referenzhandbuch: *Systemsoftware für M7-300/400,*
System- und Standardfunktionen

/282/ Benutzerhandbuch: *Systemsoftware für M7-300/400,*
Installieren und Bedienen

/290/ Benutzerhandbuch: *ProC/C++ für M7-300/400,*
C-Programme erstellen

/291/ Benutzerhandbuch: *ProC/C++ für M7-300/400,*
Debugger für C-Programme

/292/ Benutzerhandbuch: *DDE-Server für M7-300/400,*
Installieren und Bedienen

/293/ Elektronisches Handbuch: *Systemsoftware für M7-300/400,*
Ladbare Treiber erstellen

Regelung

/349/ Benutzerhandbuch: *SIMATIC 7,*
Modular PID Control

/350/ Benutzerhandbuch: *SIMATIC 7,*
Standardregelung

/351/ Benutzerhandbuch: *SIMATIC 7,*
Fuzzy-Control

/352/ J. Gißler, M. Schmid: Vom Prozeß zur Regelung. Analyse, Entwurf, Realisierung in der Praxis. Siemens AG. ISBN 3-8009-1551-0.

NCM

/500/ Handbuch: *SIMATIC NET,*
NCM S7 für Industrial Ethernet

/501/ Handbuch: *SIMATIC NET*
NCM S7 für PROFIBUS

PG

/598/ Referenzhandhandbuch: *SIMATIC PC FI25,*

/599/ Benutzerhandbuch: *SIMATIC PC FI25*

/600/ Referenzhandbuch: *SIMATIC PC RI25/RI45,*

/601/ Benutzerhandbuch: *SIMATIC PC RI25/RI45,*

/602/ Referenzhandbuch: *SIMATIC PC BI10/FI10/FI15*

/603/ Benutzerhandbuch: *SIMATIC PC BI10/FI10/FI15*

/**650**/ Referenzhandbuch: PG 720

/651/ Referenzhandbuch: PG 740

/652/ Referenzhandbuch: PG 760

Handbücher nur auf CD

/800/ *DOCPRO*
Schaltbücher normgerecht erstellen (nur auf CD)

/801/ *TeleService für S7, C7 und M7*
Fernwartung eines Automatisierungssystems (nur auf CD)

/802/ *S7-PLCSIM*
Programmtest mit simulierter S7-CPU (nur auf CD)

/803/ Referenzhandbuch: *Systemsoftware für S7-300/400*
STEP 7 Standardfunktionen Teil 2 (nur auf CD)

Glossar

Client-Server-Prinzip

Bei einem Datenaustausch nach dem Client-Server-Prinzip vergibt der Client immer die Kommunikationsaufträge und der Server führt sie aus.

Device

Treiberprogramm, Ein-/Ausgabeoperation werden von RMOS über spezielle Programme, Treiberprogramme oder kurz Treiber genannt, und deren Untereinheiten, den Units, abgewickelt. Die Treiber werden vom Betriebssystem durch eine Kennung identifiziert.

Hauptverzeichnis

Dateien werden im HSFS in Verzeichnissen abgelegt. Verzeichnisse bilden im HSFS eine hierarchische Struktur. Das Hauptverzeichnis, auch Root-Directory genannt, ist das oberste Verzeichnis eines Datenträgers. Es gibt pro Datenträger (Partition) nur ein Hauptverzeichnis.

Job

Programme und Kommandos, die vom CLI (Kommandozeileninterpreter des RMOS, engl. **C**ommand **L**ine **I**nterpreter) gestartet wurden.

MS-DOS-Systemspeicher

Der RAM-Speicher zwischen 0 und 640 Kbyte, der unter Kontrolle von MS-DOS stehen kann.

PIC

Programmable **I**nterrupt **C**ontroller, programmierbarer Interrupt-Controller

Platzhalterzeichen

Auch Wildcard, Joker-Zeichen genannt. Sonderzeichen als Auslassungszeichen:

”*”	steht für eine Gruppe von Buchstaben oder Ziffern,
”?”	steht für ein einzelnes alphanumerisches Zeichen.

Queue

Warteschlange

SMR

System **M**emory **R**esource, Systemspeicherblock, wird vom Kern zur internen Verwaltung angelegt.

SVC

Super**v**isor **C**all, Systemaufruf

TCB

Task **C**ontrol **B**lock, Task-Steuerblock, Tabelle mit den aktuellen, dynamisch veränderbaren Daten zur Steuerung einer Task.

TCD

Task **C**ontrol **D**ata, Task-Steuerdaten, Tabelle mit den Vorgabewerten, wird bei statischen Tasks durch die Konfigurierung, bei dynamischen Task durch den Lader angelegt.

Unit

Ein-/Ausgabegerät. Ein oder mehrere Units werden durch Treiber angesprochen. Welche Units ein Treiber ansprechen kann, wird in der Treiberkonfigurierung festgelegt. Units werden vom Betriebssystem anhand einer Kennung, identifiziert.

Unterverzeichnis

Alle Verzeichnisse auf einem Datenträger, die nicht das Hauptverzeichnis sind, sind Unterverzeichnisse (engl. Subdirectory). Ihnen muß ein Name gegeben werden, der innerhalb des Verzeichnisses, in dem sie angelegt wurden einmalig ist.

Index

Zahlen

3964–Treiber, 7-4

A

Ablaufumgebung , CLI, 4-4
Adresse, C-3
Alarmstatus, D-13
Argument, C-3
ASM, C-7
Ausdruck, C-3

B

BASE, 6-8, C-9
Baugruppen-Identifikation, D-6
Baugruppendiagnosedaten, D-28
Baugruppendiagnoseinfo, D-27
Baugruppenstörung, E-2
Baugruppentypkennung, D-4
Baugruppenzustand anzeigen, M7-300/400, 4-17
Baugruppenzustandsinformation, D-22
Bausteintypen, D-11
Begrenzer, C-4
Betriebsmittel, RMOS–DOS, A-12
Betriebssystemversion, 3-3
Betriebssytem installieren, Laufwerke, 3-5
Betriebszustand, D-14
Betriebszustandübergang, D-14
bootfile, B-33
Bootlaufwerk, 3-7
Bootmedium, 3-6
Bootpartition, 3-7
Bootvorgang, A-8
  RMOS–DOS, A-12
Breakpoint, 6-5, C-6, C-47, C-51
Breakpoint–Kontext, 6-5, C-3, C-15, C-21, C-25, C-39, C-47, C-49, C-51, C-55
Breakpoint–Nummer, C-47
BREAKS, C-10

C

CALCULATE, C-11
CALL, C-12
CANCEL, B-5
CATALOG, C-16
CD, B-6
CFC-Programm, 1-1
CHANGE, C-13
CHGKBD, B-7
CLI
  Drucken, 4-9
  Editieren von Kommandos, 4-6
  Gestartete Kommandos und Programme, 4-6
  Starten, 4-5
  Umlenkung von Eingaben und Ausgaben, 4-8
CLI Fehlermeldungen, B-43
CONT, C-14
COPY, B-8
CPRI, B-10
CPU-eigene SZL, D-29
CPU-Merkmale, D-7
CPU–Meldungen, M7-300/400, 4-17
CPUREG, C-15

D

DATE, B-11
Dateiverwaltungssystem, A-8, A-13
  HSFS, 1-8
  RMOS–DOS, A-13
Datenlaufwerk, 3-7
Datenpartition, 3-7
Datensicherheit, 3-7
Debugregister, 6-5, C-47
Debugregister–Breakpoint, C-6, C-47
Deinstallation, 2-2
DEL, B-12
DEVICE, B-13

Diagnosedaten, E-1
Aufbau, E-1
der CPU, D-2
der Signalbaugruppen, D-2
Inhalt, E-1
Diagnosepuffer, D-2, D-24, D-26
DIR, B-16, C-16
DISMOUNT, B-17
DISPLAY, C-19
DOS–Task, RMOS–Schnittstelle, A-14
Drucken, 4-9
Durchlaufzähler, C-36, C-47

E

ECHO, B-18
EGA/VGA–Konfiguration, A-6
Einzelschritt–Modus, C-47, C-51
ERROR, B-19
Escape–Kontext, 6-5, C-39, C-40, C-55
EVALUATE, C-20
Execution–Breakpoint, C-6, C-47
EXIT, B-20, C-21
EXITK, 6-5, C-22, C-30

F

Fehler extern, E-2
Fehler intern, E-2
Fehlerbehandlung, RMOS–DOS, 1-10
Fehlermeldungen, C-58
Festplatte, Kapazität, 5-3
Festplatte partitionieren, 3-6
FILL, C-23
FORMAT, B-21
Format, C-5
Formatieren
Diskette, 5-3
Festplatte, 5-3
Memory Card, 5-2
OSD, 5-2
unter M7 RMOS32, 5-3
unter MS–DOS, 5-2, 5-9
Formatiern, M7–Zielmedium, 3-6
FREETASK, C-24
Frontstecker fehlt, E-2
FTLFORM, 5-9
M7 RMOS32, B-22

G

Geräteeinheiten, A-5
Gerätenamen, CLI, 4-9
Gleitpunktzahl, 6-8
GO, 6-5, 6-6, C-25

H

HALT, C-26
Hard–Breakpoint, C-6, C-47
Hardware, Kompatibilität, 2-3
HDPART, 5-4
HELP, B-24, C-27
Hilfsspannung fehlt, E-2

I

IN, C-28
INHIB, 6-6, C-29
Inline–Kommandos, 4-4
Installieren, 2-1
Hardwarevoraussetzungen, 2-1
Setup, 2-1
Softwarevoraussetzungen, 2-2

J

Job, 4-6

K

Kanalfehler, E-2
KILL, C-30
device, Kommando , 3-25
Kommando HDPART, 5-4
Kommunikation, Zustandsdaten, D-18
Konstante, C-4
Kontext, 6-5
Kontext , 6-3
Kontroll–Lesen, C-13, C-23

L

Ladbare Treiber, 7-3
Datei RMOS.INI, 3-25
Laufwerke, 3-5

Leistungsmerkmale, A-2
LINES, C-31
Loader–Result–Segment, C-32
LOADTASK, C-32
Lokales Laufwerk, 3-5

M

M7 RMOS32
  Installieren auf Memory Card, 3-8
  Treiber, A-8
M7 RMOS32 , Konfigurierung, A-5
M7-300/400
  Baugruppen, 1-6
  Betriebssysteme, 1-5
M7-Anwendungsprogramm
  auf Zielsystem übertragen, 4-11
  löschen, 4-15
  starten, 4-16
  übertragen über Datenträger, 4-14
  übertragen über MPI, 4-13
M7-Betriebssytem
  auswählen, 3-4
  Installieren, 3-2
M7–Zielsystem
  Bootmedium, 3-6
  Zielmedium, 3-2, 3-4
MD, B-25
Memory Card
  RAMDIR, 5-11
  Ziehen und Stecken, 5-11
MONITOR, C-34
Monitor–Modus, 6-2
MOUNT, B-26

N

Nachladbare Kommandos, 4-4
NPX, B-27

O

Optionssoftware, 1-2
OUT, C-35

P

Parameter, falsche in der Baugruppe, E-2
Parametrierung fehlt, E-2
Partitionieren, 3-6, 5-4
Partitionierung, 5-5
Partner Laufwerk, 3-5
PATH, B-28
Privilegverletzungen, RMOS–DOS, 1-10
Programme abbrechen, 4-16
Programme starten, 4-16
PROMPT, B-31

Q

QUALIFY, C-36
QUERY, C-37

R

RD, B-32
RDISK, B-33
REGS, C-39
REMAP_A, 5-12
RENAME, B-34
REPORT, C-41
Resource Reporter, C-41
RM3PMEM.SYS, A-16
RM3RESET.SYS, A-13, A-16

S

S7OFORMX, 5-9
S7OFORNX, 5-9
SCANDISK, B-35
Segment/Selektor, C-19
Segmentdescriptor, C-3
Segmentregister, C-3
Selektor, C-3
SER8250–Treiber, 7-4
SESSION, B-37
SET, B-38, C-47
Single–Step–Interrupt, 6-5
Soft–Breakpoint, C-6, C-25, C-47
Speicherbelegung, A-9
  AT–Karten, A-10
Speicherbereiche, D-9
Speicherformat, C-4, C-13, C-23
SRAM–Bereich, A-4
STACK, C-49
START, B-39, C-50
STEP, C-51
stl_load, C-32
SVC, C-52
SWITCH, C-54
SYSTAT, B-40
Systembereiche, D-10

Systemdaten, D-2
Systemkonsole, A-6
  Datei RMOS.INI, 3-25
Systemzustandsliste, D-2
  Auflistung, D-5
SZL, D-2, D-3
  Auflistung, D-5
SZL–ID, D-4

T

TASK, C-55
Task–ID, C-5
Task–Modus, 6-2
Task–Steuerblock, C-56, C-57
Task–Verwaltung, M7 RMOS32–DOS, A-13
TCB, C-56
TCD, C-57
TIME, B-41
Treiber, 7-1
Treiber entfernen, 7-4
Treiber laden
  CLI, 7-3
  Datei RMOS.INI, 7-3
Typkennung, Baugruppen–, D-4

U

Übertragen, M7-Anwendungsprogramm, 4-13
Übertragungsmedium, 4-11
Umlenkung , CLI, 4-8

V

Variable, beobachten und steuern,
  M7-300/400, 4-18
VER, B-42

W

Warmstart, RMOS–DOS, 1-10
Watchdog, A-9
Wiederholungszhler, 6-8

Z

Zahlenbasis, 6-8
Zahlendarstellung, 6-8
Ziehen/Stecken der Memory Card , 5-11
Zielmedium, 3-6
  formatieren, 3-6

An
Siemens AG
A&D AS E48
Postfach 4848
D–90327 Nürnberg

Absender:

Ihr Name: _ _ _

Anmerkungen/Vorschläge

Ihre Anmerkungen und Vorschläge helfen uns, die Qualität und Benutzbarkeit unserer Dokumentation zu verbessern. Bitte füllen Sie diesen Fragebogen bei der nächsten Gelegenheit aus und senden Sie ihn an Siemens zurück.

Geben Sie bitte bei den folgenden Fragen Ihre persönliche Bewertung mit Werten von 1 = gut bis 5 = schlecht an.

1. Entspricht der Inhalt Ihren Anforderungen? ☐
2. Sind die benötigten Informationen leicht zu finden? ☐
3. Sind die Texte leicht verständlich? ☐
4. Entspricht der Grad der technischen Einzelheiten Ihren Anforderungen? ☐
5. Wie bewerten Sie die Qualität der Abbildungen und Tabellen? ☐

Falls Sie auf konkrete Probleme gestoßen sind, erläutern Sie diese bitte in den folgenden Zeilen:

– – – –